Time-accurate anisotropic mesh adaptation for three-dimensional time-dependent problems with body-fitted moving geometries

Nicolas Barral, Géraldine Olivier, Frédéric Alauzet

Time-Accurate Anisotropic Mesh Adaptation for Three-Dimensional Time-Dependent Problems with Body-Fitted Moving Geometries

N. Barral[a], G. Olivier[a], F. Alauzet[a]

[a]*INRIA Saclay Ile-de-France, Projet Gamma3, 1, rue Honoré d'Estienne d'Orves, 91126 Palaiseau, France.*

Abstract

Anisotropic metric-based mesh adaptation has proved its efficiency to reduce the CPU time of steady and unsteady simulations while improving their accuracy. However, its extension to time-dependent problems with body-fitted moving geometries is far from straightforward. This paper establishes a well-founded framework for multiscale mesh adaptation of unsteady problems with moving boundaries. This framework is based on a novel space-time analysis of the interpolation error, within the continuous mesh theory. An optimal metric field, called ALE metric field, is derived, which takes into account the movement of the mesh during the adaptation. Based on this analysis, the global fixed-point adaptation algorithm for time-dependent simulations is extended to moving boundary problems, within the range of body-fitted moving meshes and ALE simulations. Finally, three dimensional adaptive simulations with moving boundaries are presented to validate the proposed approach.

Key words: Metric-based mesh adaptation, anisotropy, unsteady, moving geometry, moving mesh, connectivity change.

1. Introduction

Simulating complex moving geometries evolving in unsteady flows in three dimensions, which is more and more required by industry, still remains a challenge because it is very time consuming. To reduce the CPU time of numerical simulations while preserving their accuracy, anisotropic metric-based mesh adaptation has already proved its efficiency for steady problems [5, 37, 38, 46], and appears as a promising way to reduce the complexity of such simulations. However, its extension to the unsteady case with body-fitted moving geometries is not straightforward. Indeed, time-dependent simulations combine the difficulties arising from unsteadiness and geometrical complexity: global time-step driven by the mesh smallest height, evolution of physical phenomena in the whole domain, solution interpolation spoiling, and also three-dimensional meshing and remeshing issues with an imposed discretized surface. The introduction of body-fitted moving geometries in this process even raises new difficulties, due to the handling of the mesh movement and the deterioration of its quality, the specific numerical schemes imposed by moving geometries as well as fluid/structure coupling and contact handling.

There exist two main branches of mesh adaptation for unsteady problems: r-adaptation and adaptive remeshing.

In the case of r-adaptation also called adaptive moving mesh methods, the mesh is continuously moved to an adapted configuration. The movement of the mesh vertices is governed by an extra equation that is strongly coupled to the underlying physics equations and that is solved at the same time. Since the mesh movement is explicitly taken into account in the equations, there are no spoiling interpolation steps. Several approaches can be found in the literature, that differ by the adaptation criteria, the way the adaptation governs the mesh movement and the way the mesh movement is coupled with the physics equations. Recent works on the topic include Moving Mesh PDEs [8, 28], and Lagrangian methods with ALE rezoning [40]. A Monge-Ampère equation has also been used to drive the adaptation and has shown interesting results in 3D in [15, 17]. However, the solution of the extra moving mesh equation is often costly in terms of CPU in 3D, and the optimality of the adapted mesh is not achieved temporally as the same number of degrees of freedom is used through the whole simulation. Moreover, it is unsure how these methods can be coupled to moving boundary problems.

The other branch consists in performing frequent adaptive remeshings. Three different approaches can be distinguished in the literature. First [30, 33, 49, 52], an isotropic mesh is adapted frequently in order to maintain the solution within refined regions and introduce a safety area around critical regions. Another approach is to use an unsteady mesh adaptation algorithm [14, 16, 20, 48, 54] based on local or global remeshing techniques and the estimation of the error every n flow solver iterations. If the error is greater than a prescribed threshold, the mesh is re-adapted. In [27], the authors combine both approaches mentioned above. Local adaptive remeshing enabling the construction of anisotropic meshes has been also considered. In this case [46, 50], the mesh is frequently adapted in order to guarantee that the solution always evolves in refined regions. All these approaches involve a large number of mesh adaptations leading to a large increase in CPU time due to the generation of many meshes[1], and

[1]The generation of a mesh is usually more costly than a few flow solver time-steps.

the numerous IOs and database initializations[2]. This large number of mesh adaptations also introduces unquantified errors due to the transfer of the solution from the old mesh to the new one. In the context of conservative equations, this can result in a time-shift of the solution if a non-conservative solution transfer is considered [2]. Moreover, these methods can result in a time shift between the mesh and the solution, since the adapted mesh is generated at one time-step, from the solution at that time-step, and is kept for the next few time-steps. Finally, none of them consider the inherent non-linear nature of the mesh adaptation problem: the convergence of the mesh adaptation process is never addressed and therefore obtaining the optimal mesh cannot be expected.

A first answer to these issues has already been proposed [3, 22, 25, 41]. It is based on a control of the space-time interpolation error in L^∞ norm, the subdivision of the time interval into large sub-intervals on which the mesh is kept constant and on a *local* fixed-point algorithm to address the prediction of the solution and the convergence of the non-linear mesh adaptation problem. Since then, multiscale anisotropic mesh adaptation [5, 38] (*i.e.,* the control of the interpolation error in L^p norm) and goal-oriented anisotropic mesh adaptation [37] (*i.e.,* the control of a goal-oriented error estimate in L^1 norm) have proved to be a lot more efficient for steady CFD computations. Therefore, it seems relevant to extend these approaches to time-dependent simulations. Time-accurate multiscale and goal-oriented anisotropic mesh adaptation have been derived in [6] and [13], respectively, but for fixed meshes. They are based on a space-time interpolation error analysis and a *global* fixed-point mesh adaptation algorithm. The present paper extends the time-accurate multiscale anisotropic mesh adaptation of [6] to the case of moving geometries with dynamic meshes.

Three classes of methods exist to deal with moving geometries: immersed/embedded methods, chimera/overset methods, and body-fitted methods. In this work, we select the last class. If very frequent remeshings are performed (each n flow solver time-steps), the movement of the geometries and thus of the meshes can be dealt with by the remeshings [18, 21]. But, this leads to the issues stated above. In our case, the adaptive remeshings are much less frequent as only one adapted mesh is generated for each sub-interval, so the handling of the moving geometries is crucial, both from the purely moving mesh point of view and from the solver point of view. Indeed, algorithmically (to keep a simple mesh adaptation scheme) and theoretically (to derive the error analysis), it is fundamental to only remesh when it is required by the adaptation, *i.e.,* for each sub-interval, and to never remesh because the mesh becomes too distorted due to the mesh deformation. Many works exist on mesh deformation strategies and Arbitrary-Lagrangian-Eulerian (ALE) numerical schemes [12, 26, 34, 43, 47], but generally they are not able to handle large geometry displacement without any remeshing throughout the simulation. Here, we consider the connectivity-change moving mesh algorithm proposed in [1, 11] which proves to be very efficient to handle any 3D geometry displacements thanks to mesh optimizations based on connectivity changes that manage efficiently any shearing inside the mesh deformation.

Regarding adaptive strategies for moving mesh simulations, only a few attempts can be found in the literature among which [18, 26, 29, 31, 32, 51]. As impressive as they can be, these results nevertheless still suffer from some of the weaknesses described above, *i.e.,* mainly very frequent remeshing and spoiling interpolation stages. Moreover, there is no well-established framework for the space-time error estimate and for the consideration of the dynamic of the mesh. The contribution of this paper is to propose an extension of the time-accurate multiscale anisotropic mesh adaptation to moving-geometry problems, and to demonstrate numerically that three-dimensional moving mesh adaptation simulations can actually be run with this algorithm.

To this aim, we first (Section 2) recall the extension of the space-time error analysis of [13] to time-accurate multiscale anisotropic mesh adaptation, which provides a control of the space-time interpolation error in L^p norm and is thoroughly detailed in [6]. Section 3 describes the global fixed-point mesh adaptation algorithm and its practical implementation. The extension of the space-time analysis and of the mesh adaptation algorithm to moving mesh simulations is then addressed in Sections 4 and 5. The contributions of this work are the following:

- propose an ALE metric field formulation which takes into account the displacement of the mesh. This ALE metric field formulation is validated on several analytic examples by analyzing the quality of the resulting meshes and by performing a convergence study of the error.
- extend the space-time interpolation error analysis to dynamic meshes using the ALE metric field formulation. The idea is to match instantaneous interpolation error using the ALE metric field in order to map the error onto the initial continuous mesh configuration, this removes the dependence of the continuous mesh in time.
- modify the connectivity-change moving mesh algorithm [1] to take into account anisotropic adapted dynamic meshes. In particular, a dynamic metric field is considered to perform the mesh optimizations.
- demonstrate that body-fitted ALE numerical simulations involving large displacement of complex geometries and coupled with anisotropic mesh adaptation is now achievable in a very robust and efficient way. To this end, many numerical examples of 3D unsteady adaptive moving mesh simulations are presented and analyzed (Section 6).

[2]Unless a flow solver with dynamic data fully coupled with the adaptive remesher is used.

2. Time-accurate multiscale anisotropic mesh adaptation for unsteady problems

This section first recalls the context of metric-based mesh adaptation, then a space-time interpolation error analysis is presented. Presenting the context and the time-accurate anisotropic mesh adaptation framework is essential to understand the analysis with dynamic meshes. Note that the results are only presented in dimension three.

2.1. Metric-based generation of anisotropic adapted meshes

Metric-based generation of anisotropic adapted meshes uses the notion of Riemannian metric space [23, 35, 36]. For a computational domain $\Omega \subset \mathbb{R}^3$, a Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is a spatial metric field that defines at any point of Ω a metric tensor $\mathcal{M}(\mathbf{x})$, e.g. a 3×3 symmetric positive definite matrix. It is then possible for a mesh generator to work, *i.e.,* to evaluate all geometric quantities, in this Riemannian metric space instead of working in the canonical Euclidean space. In a Riemannian metric space, the dot product is defined locally by metric tensor $\mathcal{M}$: $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{M}} = \langle \mathbf{u}, \mathcal{M}\mathbf{v} \rangle$ for $(\mathbf{u}, \mathbf{v}) \in \mathbb{R}^3 \times \mathbb{R}^3$. Thus, the computation of geometric quantities requires integral formulae to take into account the variation of the metric field. In that case, the length of edge $\mathbf{ab}$ is computed using the straight line parameterization $\gamma(t) = \mathbf{a} + t\,\mathbf{ab}$, where $t \in [0, 1]$:

$$\ell_{\mathcal{M}}(\mathbf{ab}) = \int_0^1 \|\gamma'(t)\|_{\mathcal{M}}\, \mathrm{d}t = \int_0^1 \sqrt{\mathbf{ab}^T\, \mathcal{M}(\mathbf{a} + t\,\mathbf{ab})\, \mathbf{ab}}\, \mathrm{d}t\,,$$

and the volume of element K is:

$$|K|_{\mathcal{M}} = \int_K \sqrt{\det \mathcal{M}(\mathbf{x})}\, \mathrm{d}\mathbf{x}\,.$$

The main idea of metric-based mesh adaptation, initially introduced in [24], is to generate *a unit mesh* in the prescribed Riemannian metric space, e.g. a mesh of $\Omega \subset \mathbb{R}^3$ such that each edge has a unit length and each tetrahedron is regular (or equilateral) with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$:

$$\forall \mathbf{e},\ \ \ell_{\mathcal{M}}(\mathbf{e}) = 1 \quad \text{and} \quad \forall K,\ \ |K|_{\mathcal{M}} = \frac{\sqrt{2}}{12}\,.$$

The resulting mesh in the canonical Euclidean space will be anisotropic and adapted. As it is not possible to tesselate $\mathbb{R}^3$ with the regular tetrahedron, we seek for a mesh such that all its edges have a length close to unity and such that all the elements are almost regular in the considered Riemannian metric space.

Following the duality introduced in [35, 36], a metric tensor (when it is defined point-wise) is a continuous element, and a Riemannian metric space (when it is defined all-over the domain) - also called metric field - is a continuous mesh. One or the other of these names will be used thereafter.

2.2. Space-time L^p interpolation error analysis

The space-time error analysis is based on the continuous mesh framework presented in [35, 36]. However, the multiscale mesh adaptation described in these papers only controls spatial errors. In the context of time-dependent problems, temporal errors must be controlled as well. In [13], a space-time error analysis is performed for a goal-oriented error estimate. Here, we recall the extension of this analysis to the control of the space-time interpolation error in L^p norm. In what follows, we do not account for time discretization errors but we focus on a space-time analysis of the spatial errors in unsteady simulations. In other words, we seek for the optimal space-time mesh controlling the space-time spatial discretization error.

The following assumption is made: *as an explicit time scheme is used for time advancing, the error in time is controlled by the error in space under the CFL condition.* The proof of this assumption for the 1D scalar advection equation is provided in [3] and the extension of this proof to the multi-dimensional linear advection problem is given in [44]. However, there is no proof of this fact for multi-dimensional non-linear problems. This assumption is illustrated on Sod's shock tube problem in [44] where, for explicit and implicit schemes with a CFL number less than one, we observe that no error in time appears while for implicit schemes with a CFL number larger than one large error in time appears. Therefore, if implicit schemes with arbitrary large time-steps are considered, an adaptive time-stepping method is required to control the error in time such as the analysis proposed in [19] for incompressible unsteady simulations and [44] for the multi-dimensional linear advection problem. As far as the above assumption is true, the spatial interpolation error is a good measure of the total space-time error of the discretized unsteady system.

The space-time L^p interpolation error analysis is thoroughly detailed in [6]. However, it is absolutely essential to understand well the reasoning to understand its extension to the moving mesh case, which is why some of it is recalled in the present paper. Our goal is to solve an unsteady PDE which is set in the computational space-time domain $\mathcal{Q} = \Omega \times [0, T]$ where $\Omega \subset \mathbb{R}^3$ is the spatial domain and T is the (positive) maximal time. An essential ingredient of our discretization and of our analysis is the

element-wise linear interpolation operator. We define our working functional space which is the set of measurable functions that are continuous with square integrable gradients:

$$V = \left[H^1(\Omega) \cap C(\bar{\Omega})\right], \quad \text{and} \quad \mathcal{V} = H^1\{[0,T];V\}.$$

We assume that Ω is covered by a finite-element partition made of simplicial elements denoted K. The mesh, denoted by $\mathcal{H}$, is the set of the elements. Let us introduce the following approximation spaces:

$$V_h = \left\{\varphi_h \in V \;\middle|\; \varphi_{h|K} \text{ is affine } \forall K \in \mathcal{H}\right\}, \quad \text{and} \quad \mathcal{V}_h = H^1\{[0,T];V_h\} \subset \mathcal{V}.$$

Let Π_h be the usual $\mathbb{P}^1$ projector:

$$\Pi_h : \; V \to \; V_h \;\text{ such that }\; \Pi_h\varphi(\mathbf{x}) = \varphi(\mathbf{x}), \;\forall\, \mathbf{x} \text{ vertex of } \mathcal{H}.$$

We extend it to time-dependent functions:

$$\Pi_h : \; \mathcal{V} \to \; \mathcal{V}_h \;\text{ such that }\; (\Pi_h\varphi)(t) = \Pi_h(\varphi(t)), \;\forall\, t \in\; [0,T].$$

The problem of mesh adaptation consists in finding the space-time mesh $\mathcal{H}$ of $\mathcal{Q}$ that minimizes the space-time linear interpolation error $u - \Pi_h u$ in L^p norm, for a given sensor function u and for a given number of space-time mesh vertices N_{st}. The problem is thus stated in an *a priori* way:

$$\text{Find } \mathcal{H}_{opt} \text{ having } N_{st} \text{ space-time vertices such that } \mathbf{E}_{L^p}(\mathcal{H}_{opt}) = \min_{\mathcal{H}} \|u - \Pi_h u\|_{L^p(\Omega_h \times [0,T])}. \tag{1}$$

As it, this problem is a global combinatorial problem which turns out to be intractable in practice. This ill-posed problem can be reformulated in the continuous mesh framework [35, 36]. In this framework, we propose the following continuous model of a mesh. A continuous mesh $\mathbf{M}$ of a domain Ω is identified to a Riemannian metric space $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ that, at each point $\mathbf{x}$ of Ω, prescribes a density, a set of anisotropy directions and the stretching along these directions, this information being locally contained in metric tensor $\mathcal{M}(\mathbf{x})$. The spatial size of the continuous mesh is given by its spatial complexity: $C(\mathbf{M}) = \int_\Omega \sqrt{\det \mathcal{M}(\mathbf{x})}\mathrm{d}\mathbf{x}$ which is the continuous counterpart of the number of vertices. Continuous mesh $\mathbf{M}$ defines a class of equivalence of discrete meshes, which are all unit meshes with respect to $\mathbf{M}$. We also define a continuous model of the linear interpolation operator Π_h denoted $\pi_{\mathcal{M}}$. It is then possible to recast (1) into a well-posed continuous global optimization problem of finding the optimal space-time continuous mesh minimizing the space-time continuous interpolation error in L^p norm, for a given sensor u and for a given space-time complexity $\mathcal{N}_{st}$:

$$\text{Find } \mathbf{M}_{L^p} = (\mathcal{M}_{L^p}(\mathbf{x},t))_{(\mathbf{x},t)\in\mathcal{Q}} \;\text{ such that }\; \mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = \min_{\mathbf{M}} \|u - \pi_{\mathcal{M}} u\|_{L^p(\Omega\times[0,T])}, \tag{2}$$

under the space-time complexity constraint:

$$C_{st}(\mathbf{M}) = \int_0^T \tau(t)^{-1} \left(\int_\Omega \sqrt{\det(\mathcal{M}(\mathbf{x},t))}\, \mathrm{d}\mathbf{x} \right) \mathrm{d}t = \mathcal{N}_{st}. \tag{3}$$

The continuous mesh space-time complexity $\mathcal{N}_{st}$ enables the user to control the level of accuracy of the mesh, and thus, to implicitly control the number of space-time vertices of the resulting discrete mesh. In its definition, $\tau(t)$ is the time-step used at time t of interval $[0,T]$ and $\sqrt{\det(\mathcal{M}(\mathbf{x},t))}$ represents the continuous mesh local density. The continuous space-time error model can be written as follows (see [35]):

$$\mathbf{E}_{L^p}(\mathbf{M}) = \left(\int_0^T \int_\Omega \mathrm{trace}\left(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t) |H_u(\mathbf{x},t)| \mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t) \right)^p \mathrm{d}\mathbf{x}\, \mathrm{d}t \right)^{\frac{1}{p}}. \tag{4}$$

where H_u is the Hessian of sensor u. To find the optimal space-time continuous mesh (see [6, 13]), Problem (2-3) is solved in two steps: first a spatial minimization is done for a fixed t, then a temporal minimization is performed.

Spatial minimization for a fixed t. At time t, we seek for the optimal continuous mesh $\mathbf{M}_{L^p}(t)$ which minimizes the instantaneous interpolation error $\widetilde{\mathbf{E}}_{L^p}$ in which the integral in time of Relations (3) and (4) was removed. Similarly to [35], solving the optimality conditions provides the *optimal instantaneous continuous mesh in L^p norm* $\mathbf{M}_{L^p}(t) = (\mathcal{M}_{L^p}(\mathbf{x},t))_{\mathbf{x}\in\Omega}$ at time t defined by:

$$\mathcal{M}_{L^p}(\mathbf{x},t) = \mathcal{N}(t)^{\frac{2}{3}} \left(\int_\Omega (\det |H_u(\bar{\mathbf{x}},t)|)^{\frac{p}{2p+3}} \mathrm{d}\bar{\mathbf{x}} \right)^{-\frac{2}{3}} (\det |H_u(\mathbf{x},t)|)^{-\frac{1}{2p+3}} \, |H_u(\mathbf{x},t)|. \tag{5}$$

We set: $\mathcal{K}(t) = \displaystyle\int_\Omega (\det |H_u(\bar{\mathbf{x}},t)|)^{\frac{p}{2p+3}} \mathrm{d}\bar{\mathbf{x}}$. The corresponding *optimal instantaneous interpolation error in L^p norm* at the power p at time t is:

$$\widetilde{\mathbf{E}}_{L^p}(\mathbf{M}_{L^p}(t)) = 3^p\, \mathcal{N}(t)^{-\frac{2p}{3}} \left(\int_\Omega (\det |H_u(\mathbf{x},t)|)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x} \right)^{\frac{2p+3}{3}} = 3^p\, \mathcal{N}(t)^{-\frac{2p}{3}}\, \mathcal{K}(t)^{\frac{2p+3}{3}}. \tag{6}$$

Temporal minimization. To complete the resolution of optimization Problem (2-3), we perform a temporal minimization. We need to find the optimal time law $t \to \mathcal{N}(t)$ for the instantaneous mesh size. We consider the case where the time-step τ is specified by the user as a function of time $t \to \tau(t)$. A similar analysis is done in [6] to deal with the case of an explicit time advancing solver subject to Courant time-step condition. After the previous spatial minimization, optimization Problem (2-3) becomes: find space-time continuous mesh $\mathbf{M}_{L^p}$ such that:

$$(\mathbf{E}_{L^p}(\mathbf{M}_{L^p}))^p = \min_{\mathbf{M}} \int_0^T \widetilde{\mathbf{E}}_{L^p}(\mathbf{M}_{L^p}(t))\,\mathrm{d}t = \min_{\mathbf{M}}\ 3^p \int_0^T \mathcal{N}(t)^{-\frac{2p}{3}}\,\mathcal{K}(t)^{\frac{2p+3}{3}}\,\mathrm{d}t \quad \text{under constraint} \quad \int_0^T \tau(t)^{-1}\mathcal{N}(t)\,\mathrm{d}t = \mathcal{N}_{st}\,.$$

After computations detailed in [6], the expression of the optimal space-time continuous mesh $\mathbf{M}_{L^p} = (\mathcal{M}_{L^p}(\mathbf{x},t))_{(\mathbf{x},t)\in Q}$ for a prescribed time-step is:

$$\mathcal{M}_{L^p}(\mathbf{x},t) = \mathcal{N}_{st}^{\frac{2}{3}} \left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}}\,\mathcal{K}(t)\mathrm{d}t \right)^{-\frac{2}{3}} \tau(t)^{\frac{2}{2p+3}} \left(\det |H_u(\mathbf{x},t)|\right)^{-\frac{1}{2p+3}} |H_u(\mathbf{x},t)|\,. \tag{7}$$

The following optimal error is obtained:

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = 3\,\mathcal{N}_{st}^{-\frac{2}{3}} \left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}}\,\mathcal{K}(t)\mathrm{d}t \right)^{\frac{2p+3}{3p}}\,. \tag{8}$$

2.3. Space-time L^p error analysis with time sub-intervals

The previous analysis provides the optimal adapted meshes for each time level, it thus requires the mesh to be adapted at each flow solver time-step which is inconceivable in practical applications. In [3, 6], the use of a coarse adapted discretization of the time axis is proposed. The basic idea consists in splitting the simulation time frame $[0,\,T]$ into n_{adap} adaptation sub-intervals:

$$[0,\,T] \;=\; [0 = t^1,\,t^2] \cup \ldots \cup [t^i,\,t^{i+1}] \cup \ldots \cup [t^{n_{adap}},\,t^{n_{adap}+1} = T]\,,$$

and to keep the same adapted spatial mesh for each time sub-interval. On each sub-interval, the mesh is adapted to control the solution accuracy from t^i to t^{i+1}. Consequently, the time-dependent simulation is performed with n_{adap} different adapted meshes. This drastically reduces the number of meshes generated during the simulation, hence the number of solution transfers. Moreover, the flow solver performs many iterations (hundreds to thousands) on the same fixed spatial mesh. This provides a first answer to the adaptation of the whole space-time mesh, the spatial mesh being kept constant for each sub-interval when the global space-time mesh is visualized.

In the following, we extend the previous error analysis to the fixed-point unsteady mesh adaptation algorithm context where the simulation time interval [0,T] is split into n_{adap} *sub-intervals* $[t^i, t^{i+1}]$ for $i = 1, .., n_{adap}$. Each spatial continuous mesh $\mathbf{M}^i$ is kept constant during each sub-interval $[t^i, t^{i+1}]$, thus it has no more dependency in time. As previously, a spatial minimization problem is solved first, followed by a temporal minimization resolution (see [6, 13] for details of the demonstration).

Spatial minimization for a given sub-interval. Given the continuous mesh spatial complexity $\mathcal{N}^i$ for the single adapted mesh used during sub-interval i, we seek for the optimal continuous mesh $\mathbf{M}^i_{L^p} = (\mathcal{M}^i_{L^p}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ solution of the following problem:

$$\begin{aligned} \mathbf{E}^i_{L^p}(\mathbf{M}^i_{L^p}) \;&=\; \min_{\mathbf{M}^i} \int_{t^i}^{t^{i+1}} \int_\Omega \mathrm{trace}\left((\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x})\,|H_u(\mathbf{x},t)|\,(\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}) \right)^p \mathrm{d}\mathbf{x}\,\mathrm{d}t \\ &=\; \min_{\mathbf{M}^i} \int_\Omega \mathrm{trace}\left((\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x})\,\mathbf{H}^i_u(\mathbf{x})\,(\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}) \right)^p \mathrm{d}\mathbf{x} \quad \text{such that} \quad C(\mathbf{M}^i) = \mathcal{N}^i\,, \end{aligned} \tag{9}$$

where matrix $\mathbf{H}^i_u$ on the sub-interval is defined by:

$$\mathbf{H}^i_u(\mathbf{x}) = \int_{t^i}^{t^{i+1}} |H_u(\mathbf{x},t)|\,\mathrm{d}t\,. \tag{10}$$

This comes to moving the integral over time into the trace in Expression (9). This can be done because $\mathbf{M}^i$ does not depend anymore on t, *i.e.,* the mesh is static. Note that this cannot extend directly when meshes become dynamic. The spatial minimization gives the solution of Problem (9) where the expressions of the optimal continuous mesh and of the optimal error are identical to Relations (5) and (6) where H_u is substituted with $\mathbf{H}^i_u$.

Temporal minimization. Again, we consider the case where the time-step τ is specified by a function of time. After the spatial minimization, the temporal minimization problem becomes:

$$(\mathbf{E}_{L^p}(\mathbf{M}_{L^p}))^p = \min_{\mathbf{M}} \sum_{i=1}^{n_{adap}} \mathbf{E}^i_{L^p}(\mathbf{M}^i_{L^p}) \quad \text{such that} \quad \sum_{i=1}^{n_{adap}} \mathcal{N}^i \left(\int_{t^i}^{t^{i+1}} \tau(t)^{-1} \mathrm{d}t \right) = \mathcal{N}_{st} \, .$$

After the temporal minimization, the expression of the optimal continuous mesh $\mathbf{M}_{L^p} = \{\mathbf{M}^i_{L^p}\}_{i=1,..,n_{adap}} = \left\{(\mathcal{M}^i_{L^p}(\mathbf{x}))_{\mathbf{x}\in\Omega}\right\}_{i=1,..,n_{adap}}$ and error are:

$$\mathcal{M}^i_{L^p}(\mathbf{x}) = \mathcal{N}_{st}^{\frac{2}{3}} \Big(\sum_{j=1}^{n_{adap}} \mathcal{K}^j \Big(\int_{t^j}^{t^{j+1}} \tau(t)^{-1} \mathrm{d}t \Big)^{\frac{2p}{2p+3}} \Big)^{-\frac{2}{3}} \Big(\int_{t^i}^{t^{i+1}} \tau(t)^{-1} \mathrm{d}t \Big)^{-\frac{2}{2p+3}} (\det \mathbf{H}^i_u(\mathbf{x}))^{-\frac{1}{2p+3}} \; \mathbf{H}^i_u(\mathbf{x}) \tag{11}$$

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = 3 \, \mathcal{N}_{st}^{-\frac{2}{3}} \Big(\sum_{i=1}^{n_{adap}} \mathcal{K}^i \Big(\int_{t^i}^{t^{i+1}} \tau(t)^{-1} \mathrm{d}t \Big)^{\frac{2p}{2p+3}} \Big)^{\frac{2p+3}{3p}} \, , \tag{12}$$

where $\mathcal{K}^i = \displaystyle\int_\Omega \Big(\det \mathbf{H}^i_u(\mathbf{x}) \Big)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}$.

3. Global fixed-point mesh adaptation algorithm

In [3], a local transient fixed-point mesh adaptation algorithm for time-dependent simulations has been proposed. However, this algorithm cannot be used here as the computation of the optimal continuous mesh given by Relation (11) involves a global normalization term which requires the knowledge of quantities over the whole simulation time frame. Thus, the complete simulation must be performed before evaluating any space-time continuous mesh. To this end, we consider a *global* fixed-point mesh adaptation algorithm covering the whole time frame $[0, T]$. This iterative algorithm is used to converge the non-linear mesh adaptation problem, and also to predict the solution evolution.

The global fixed-point unsteady mesh adaptation algorithm is schematized in Algorithm 1 where $\mathcal{H}$, $\mathcal{S}$ and $\mathbf{M}$ denote respectively meshes, solutions and metric fields (*i.e.,* continuous meshes). $\mathbf{H}$ is the Hessian-metric given by Relation (10). Several steps of the previous algorithm are not straightforward, and need to be further detailed.

Algorithm 1 Mesh Adaptation Loop for Unsteady Flows

Initial mesh and solution$(\mathcal{H}_0, \mathcal{S}_0^0)$ and set targeted space-time complexity $\mathcal{N}_{st}$

// Fixed-point loop to converge the global space-time mesh adaptation problem
For $j = 1, n_{ptfx}$

1. *// Adaptive loop to advance the solution in time on time frame* $[0, T]$
   For $i = 1, n_{adap}$
   (a) $\mathcal{S}_0^{i,j}$ = Interpolate conservatively next sub-interval initial sol. from $(\mathcal{H}^{i-1,j}, \mathcal{S}^{i-1,j}, \mathcal{H}^{i,j})$;
   (b) $\mathcal{S}^{i,j}$ = Compute solution on sub-interval from pair $(\mathcal{S}_0^{i,j}, \mathcal{H}^{i,j})$;
   (c) $|\mathbf{H}|^{i,j}$ = Compute sub-interval Hessian-metric from sol. sample $(\mathcal{H}^{i,j}, \{\mathcal{S}^{i,j}(k)\}_{k=1,nk})$;
   EndFor
2. C^j = Compute space-time complexity from all Hessian-metrics $(\{|\mathbf{H}|^{i,j}\}_{i=1,n_{adap}})$;
3. $\{\mathbf{M}^{i,j}_{L^p}\}_{i=1,n_{adap}}$ = Compute all sub-interval unsteady metric fields $(C^j, \{|H_{\max}|^{i,j}\}_{i=1,n_{adap}})$;
4. $\{\mathcal{H}^{i,j+1}\}_{i=1,n_{adap}}$ = Generate all sub-interval adapted meshes $(\{\mathcal{H}^{i,j}, \; \mathbf{M}^{i,j}_{L^p}\}_{i=1,n_{adap}})$;

EndFor

3.1. Computation of the optimal continuous mesh

Computation of the Hessian-metric. The optimal L^p metric involves an averaged Hessian-metric $\mathbf{H}^i_u$ on sub-interval i given by Relation (10), but it still remains to know how to compute it practically, *i.e.*, how it is discretized. The strategy adopted in [3] is to sample the solution on the time sub-interval. More precisely, n_k solutions equally distributed on the sub-interval time frame are saved, including the initial solution at t^i and the final solution at t^{i+1}. Positive Hessian $|H_u(\mathbf{x}, t^k)|$ is evaluated for each sample. The Hessian matrices are computed using a double least-square procedure: first the gradients are computed using a linear least-square

reconstruction procedure [42], and a similar procedure is applied to recover the Hessians. The error analysis leads to write $\mathbf{H}_u^i$ as the integral over time of the Hessian matrices, and thus the following discretization is used:

$$\mathbf{H}_u^i(\mathbf{x}) = \frac{1}{2}\frac{\Delta t^i}{n_k-1}|H_u(\mathbf{x},t^i)| + \frac{\Delta t^i}{n_k-1}\sum_{k=2}^{n_k-1}|H_u(\mathbf{x},t^k)| + \frac{1}{2}\frac{\Delta t^i}{n_k-1}|H_u(\mathbf{x},t^{i+1})| = \Delta t^i\,|H^i_{\text{avg}}(\mathbf{x})|\,,$$

where $\Delta t^i = t^{i+1} - t^i$ is the sub-interval time length and $t^k = t^i + \frac{k-1}{n_k-1}\Delta t^i$.

Choice of the optimal continuous mesh. The optimal adapted mesh for each sub-interval is obtained according to analysis of Section 2.3. For the numerical results presented below, we select the optimal mesh given by Relation (11) and the following particular choices have been made:

- the space-time error is controlled in L^2 norm ($p = 2$)
- all sub-intervals have the same time length $\Delta t = T/n_{adap}$
- function $\tau : t \to \tau(t)$ is constant and equal to Δt, thus $\int_{t^i}^{t^{i+1}} \tau(t)^{-1}\mathrm{d}t = 1$.

In that case, we can just consider the time-step $\tau(t)$ constant and equal to Δt (thanks to the global normalization term) so the integral $\int_{t^i}^{t^{i+1}} \tau(t)^{-1}\mathrm{d}t$ reduces to 1 (see [6] for details). With these choices, the optimal continuous mesh $\mathbf{M}_{L^2} = \{\mathbf{M}^i_{L^2}\}_{i=1,\dots,n_{adap}} = \{(\mathcal{M}^i_{L^2}(\mathbf{x}))_{\mathbf{x}\in\Omega}\}_{i=1,\dots,n_{adap}}$ simplifies to:

$$\mathcal{M}^i_{L^2}(\mathbf{x}) = \mathcal{N}_{st}^{\frac{2}{3}}\left(\sum_{j=1}^{n_{adap}}\Big(\int_\Omega (\det|\mathbf{H}^j_u(\mathbf{x})|)^{\frac{2}{7}}\mathrm{d}\mathbf{x}\Big)\right)^{-\frac{2}{3}}\Big(\det|\mathbf{H}^i_u(\mathbf{x})|\Big)^{-\frac{1}{7}}\,|\mathbf{H}^i_u(\mathbf{x})|\,. \tag{13}$$

In that case, as we assume that theoretically one time-step is done by sub-interval, $\mathcal{N}_{avg} = \mathcal{N}_{st}/n_{adap}$ represents the average spatial complexity by sub-interval. We do not prescribe the temporal complexity, *i.e.,* we do not control the number of time-steps done at each sub-interval.

In practice, the user prescribes the number of sub-intervals n_{adap} and the sub-interval average spatial complexity $\mathcal{N}_{avg}$ leading to a total space-time complexity of

$$\mathcal{N}_{st} = n_{adap} \times \mathcal{N}_{avg}\,. \tag{14}$$

This prescription is the theoretical complexity. In practice, the total number of space-time vertices N_{st} of the simulation discrete meshes is directly proportional to the prescribed total space-time complexity $N_{st} = c\ \mathcal{N}_{st}$, where coefficient c depends on the geometry of the problem, the mesh gradation, and the local remesher. The total number of space-time vertices of the simulation run on n_{adap} adapted meshes is computed as follow:

$$N_{st} = \sum_{i=1}^{n_{adap}} n^i_{iter} \times N^i\,,$$

where the time discretization corresponds to the number of time-steps of the flow solver at each sub-interval, e.g. n^i_{iter} is the number of time-steps for the i^{th} sub-interval, and N^i is the number of spacial vertices for the i^{th} sub-interval.

Comments on parameters $\mathcal{N}_{st}$, $\mathcal{N}_{avg}$ and n_{adap}. A very detailed analysis and convergence study of these parameters has been done in [6]. The total space-time complexity $\mathcal{N}_{st}$ can be increased by fixing n_{adap} and increasing $\mathcal{N}_{avg}$ or by fixing $\mathcal{N}_{avg}$ and increasing n_{adap}. The finer adapted meshes obtained with these two approaches have the same theoretical complexity but increasing n_{adap} leads to a mesh having less space-time vertices than increasing $\mathcal{N}_{avg}$. In other words, a better adapted space-time mesh is obtained by increasing n_{adap}. Therefore, when convergence studies are performed, the number of sub-intervals should be increased to increase the space-time complexity in order to obtain a better rate of convergence.

3.2. Matrix-free $\mathbb{P}_1$-exact conservative solution transfer

Between each sub-interval, the solution needs to be transferred from the previous mesh to the next one to pursue the computation. This stage becomes critical in the context of unsteady problems and even more if a large number of transfers is performed, as the error introduced by this stage can spoil the overall accuracy of the solution. In the context of the resolution by a second order numerical scheme of a PDE system of conservation laws, such as the compressible Euler system, it is crucial for the interpolation method to satisfy the following properties in order to obtain a consistent mesh adaptation scheme: (i) mass conservation, (ii) $\mathbb{P}_1$-exactness preserving the second order of the adaptive strategy and (iii) verify the maximum principle. This matrix-free $\mathbb{P}_1$-exact conservative solution transfer method, and its properties, is thoroughly described in [2].

3.3. The remeshing step

The remeshing step is also crucial in the adaptation process, as a poorly adapted mesh or a mesh with bad quality elements will impact the accuracy of the solution and spoil the efforts on all other parts of the process. In this paper, the remesher `Amg` [39] was used. `Amg` belongs to the class of 3D anisotropic local remeshers that aim at generating a unit mesh with respect to a prescribed metric field. Its main particularity is to adapt the volume and the surface mesh in a coupled way so that a valid 3D mesh is always guaranteed in output. It uses a unique cavity-based operator (that generalizes standard operators: point insertion, edge removal, edges swapping, point smoothing). This operator is governed by dedicated metric-based quality functions. This allows us to reach a good balance between high level of anisotropy, necessary to capture the physical features of the solution, and mesh quality, necessary to ensure the stability and accuracy of the numerical scheme. `Amg` has several enhancements designed for CFD computations, including explicit control and optimization of tetrahedra height to ensure a maximal time-step for unsteady simulations, and surface mesh re-projection based either on CAD or on background discrete meshes.

3.4. The ALE flow solver

We consider the compressible Euler equations for a Newtonian fluid in their Arbitrary-Lagrangian-Eulerian (ALE) formulation. The ALE formulation allows the equations to take arbitrary motion of the mesh into account. Assuming that the fluid is a perfect gas and that there is no thermal diffusion, the ALE formulation of the Euler equations is written for any arbitrary closed volume $C(t)$ of boundary $\partial C(t)$ moved with mesh velocity $\boldsymbol{w}$:

$$\frac{d}{dt}\left(\int_{C(t)} \mathbf{W}\mathrm{d}\mathbf{x}\right) + \int_{\partial C(t)} \left(\mathcal{F}(\mathbf{W}) \cdot \mathbf{n} - \mathbf{W}(\boldsymbol{w} \cdot \mathbf{n})\right) \mathrm{d}\mathbf{s} \quad = \quad \int_{C(t)} \mathbf{F}_{ext}\, \mathrm{d}\mathbf{x} \tag{15}$$

$$\text{where} \begin{cases} \mathbf{W} = (\rho, \rho\mathbf{u}, \rho e)^T \text{ is the conservative variables vector} \\ \mathcal{F}(\mathbf{W}) = \left(\rho\mathbf{u},\ \rho u_x\mathbf{u} + p\mathbf{e}_x,\ \rho u_y\mathbf{u} + p\mathbf{e}_y,\ \rho u_z\mathbf{u} + p\mathbf{e}_z,\ \rho\mathbf{u}h\right) \text{ is the flux tensor} \\ \mathbf{F}_{ext} = (0,\ \rho\,\mathbf{f}_{ext},\ \rho\mathbf{u}\cdot\mathbf{f}_{ext})^T \text{ is the contribution of the external forces} \end{cases}$$

and we have noted ρ the density of the fluid, p the pressure, $\mathbf{u} = (u_x, u_y, u_z)$ its Eulerian velocity, $q = \|\mathbf{u}\|$, ε the internal energy per unit mass, $e = 1/2\, q^2 + \varepsilon$ the total energy per unit mass, $h = e + p/\rho$ the enthalpy per unit mass of the flow, $\mathbf{f}_{ext}$ the resultant of the volumic external forces applied on the particle and $\mathbf{n}$ the outward normal to interface $\partial C(t)$ of $C(t)$. The fixed mesh case is obtained by simply taking $\boldsymbol{w} = \mathbf{0}$.

The following examples were run using our in-house flow solver, `Wolf`. The spatial discretization relies on an edge-based vertex-centered Finite Volume numerical scheme using an HLLC Riemann approximate solver to compute the numerical fluxes. A second-order scheme is derived according to a MUSCL type method with a low dissipation gradient reconstruction combined with a generalization of the Superbee limiter with three entries. The main difference when translating these schemes from the standard formulation to the ALE formulation is the addition of the mesh velocities $\boldsymbol{w}$ in the wave speeds of the Riemann problem and the re-evaluation of the finite volume cells at each time-step [10]. In this work, the temporal discretization is explicit and based on strong stability preserving Runge-Kutta (SSPRK) schemes [53]. The 5-stage order-2 SSPRK is considered with a CFL number equal to 3. The discrete geometric conservation law (DGCL) is strictly enforced at the discrete level by determining when and how geometrical parameters that appear in the fluxes should be computed [45].

3.5. Parallelization of the mesh adaptation loop

All the steps of the adaptation loop have been parallelized. Two different approaches are used for the two big parts of the loop. The solution computation and the solution transfer procedure are parallelized using a p-thread paradigm at the element loop level [4]. As regards the computation of the metrics, the metric gradation and the generation of the adapted meshes, a pipeline approach was used. These operations have to be done at the end of the loop, as many times as there are sub-intervals, but the operations for one sub-interval are totally independent from the operations for another. Consequently, they can all be run in parallel: if N processors are available, N metrics can be computed and their associated meshes generated simultaneously in serial on one processor.

3.6. Example of a 3D blast on the London Tower Bridge

Let us illustrate the efficiency of the time-accurate multiscale anisotropic mesh adaptation method on an example involving a very complex geometry. We consider a blast on the London Tower Bridge, whose geometry was the object of the meshing contest of the 23rd International Meshing Roundtable. A CAD description of the bridge was provided, from which an initial surface (Figure 1, left) and volume mesh were generated. This very detailed geometry is a challenge for mesh adaptation, because it results in complex pattern of the solution that the error estimate has to be able to capture, and because the adaptive remesher has to be able to preserve them and preserve the anisotropy in their vicinity.

The gas is initially at rest in a box of dimension $[-130, 130]\times[-72, 77]\times[0, 100]$. The gas is in a constant state $\mathbf{W}_0 = (1, \mathbf{0}, 2.5)$ everywhere, except for a sphere of radius 1 centered in $(3.5, 20, 10)$ containing a high density and high energy gas defined by $\mathbf{W}_{blast} = (10, \mathbf{0}, 250)$. The gas is then left to evolve freely until dimensionless time $T = 25$, and wall conditions are imposed on the boundaries. The spherical shock expands in the volume, hits the walls of the towers, and insinuates itself in any hole of the geometry, see Figure 3 (right). We observe numerous shock waves reflected by the walls that then interact with each other, and even recirculations around the pillars of the bridge. Many reflected shock waves impact the low-density bubble region located at the center of the explosion. A mushroom-shape instability develops from all these shock-bubble interactions as can be seen in Figure 2. We observe that all these phenomena - the shocks and as many of their reflections as possible as well as the smaller scale instabilities of the mushroom - are automatically captured and highly resolved thanks to the time-accurate multiscale anisotropic mesh adaptation method.

To obtain that highly resolved result, the simulation interval was split into $n_{adap} = 80$ sub-intervals and a theoretical average spatial complexity of $\mathcal{N}_{avg} = 800,000$ per sub-interval has been set. This represents a total space-time complexity $\mathcal{N}_{st}$ equal to 64 million. The density field of the solution was chosen as sensor for the adaptation. Five fixed-point iterations were used to converge the non-linear mesh adaptation problem. Practically, at the last fixed-point iteration, the discrete meshes have an average number of spatial vertices of $N_{avg} = 2,970,984$ vertices and a total of $n_{iter} = 16,353$ time-steps have been performed. The total number of space-time vertices used to compute that solution is $N_{st} = 47.7$ billion. To give an idea of the accuracy of the mesh, at $t = 3.125$, the mesh accuracy is around $1.4\, e^{-3}$, and $1.1\, e^{-3}$ at $t = 25$. The CPU time of the last fixed-point iteration (error estimate, adaptive mesh generation, solution interpolation, and flow solver) is 21 hours on 20 cores using a 2 Xeon E5-2670 v2 chip processor (10 cores at 2.5 GHz for each chip), both chips being connected by 2 QPI links with a speed of 16 GB/s.

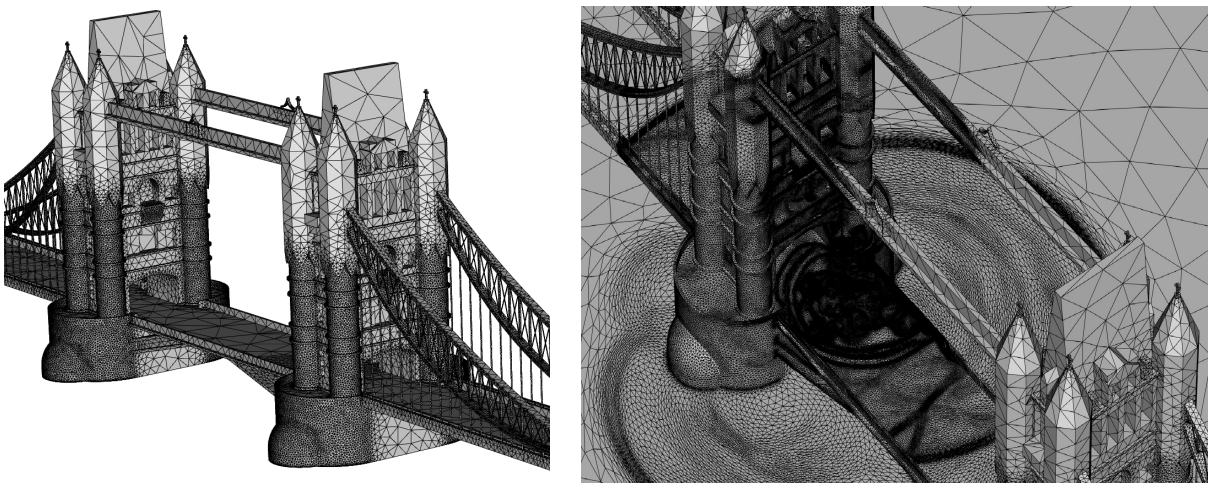

Figure 1: London tower bridge case: initial surface mesh and adapted surface mesh at dimensionless time $t = 21$.

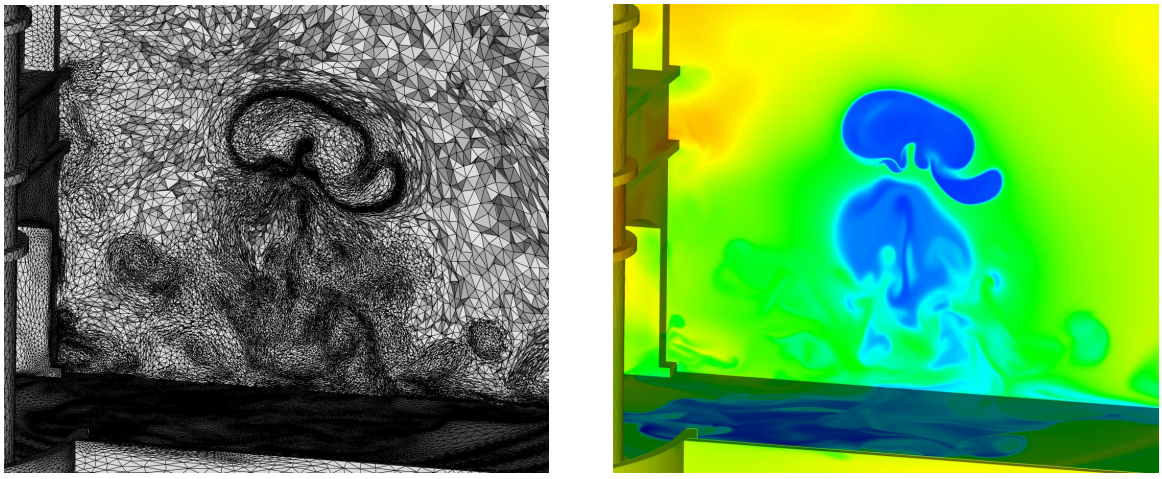

Figure 2: London tower bridge case: zoom on details of the mesh and the solution at dimensionless time $t = 21$ in the region where the explosion originated.

Snapshots of the adapted meshes and solutions are shown in Figure 3. The mesh adaptation for the whole sub-interval is clearly illustrated. Indeed, the mesh refinement along band-shaped regions, which is typical of the fixed-point algorithm, is clearly visible. These band-shaped areas correspond to the evolution zone of the physical phenomena during an adaptation sub-interval.

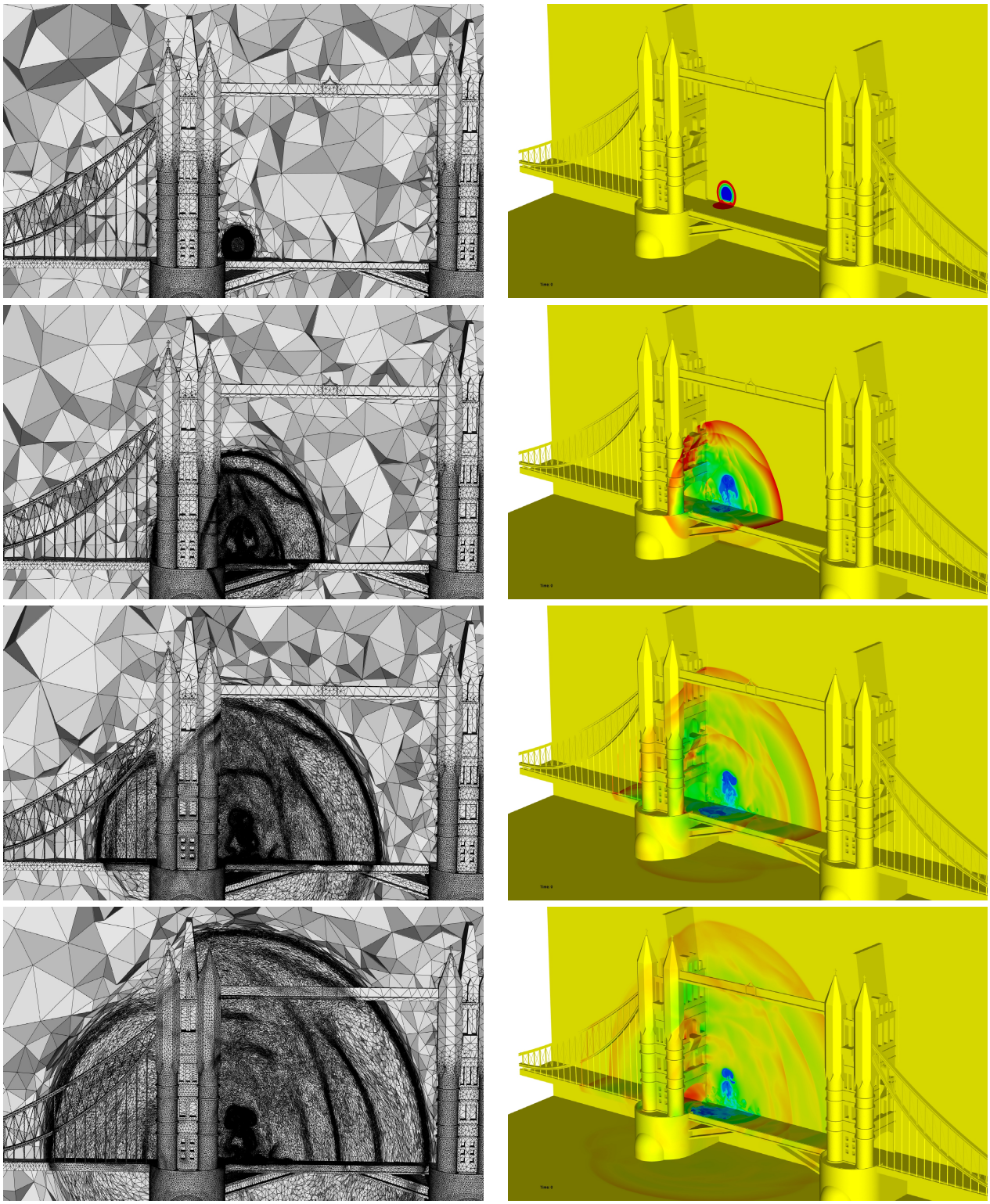

Figure 3: London tower bridge case: snapshots of adapted meshes and solution at dimensionless times $t = 0.63$, $t = 8.6$, $t = 17.2$, $t = 25$.

4. Extension of the space-time L^p error analysis to moving mesh simulations

In the previous sections, we have only considered fixed meshes in the adaptation process. Due to its efficiency in those cases, it seems natural to try to extend the global fixed-point mesh adaptation algorithm (Algorithm 1) to moving mesh simulations. In the context of body-fitted moving mesh simulations, the mesh is deformed to follow the displacement of the moving geometry inside the domain. The mesh deformation consists in prescribing a displacement to each mesh vertex by solving a PDE or by using an interpolation method where the boundary conditions are the geometry displacement. Thus, the general ideas driving the global fixed-point mesh adaptation algorithm are still valid: iterating the process to converge the couple mesh/solution and splitting the simulation time interval into sub-intervals. By design, we are only remeshing between two adaptation sub-intervals, which leads to a unique dynamic mesh on each sub-interval that is deformed because of the moving geometry. But, the previous adaptation algorithm cannot directly be extended to moving mesh simulations because it does not take into account the movement of the mesh. The error analysis of Section 2.3 from which the optimal continuous mesh is deduced, assumes that the mesh is constant in time inside a sub-interval, therefore it does not take into account the local deformation of the mesh which necessarily impacts local errors. Indeed, the mesh deformation is driven by the boundary displacement, so the mesh distortion does not necessarily follow the physical phenomena. This may in particular interfere with the adaptation of the mesh.

The previous error analysis has to be modified to take into account the movement of the mesh. As regards the discrete representation of the metric field for moving meshes, two options are available. The first approach is a Eulerian approach: a metric is associated with a fixed position in space, thus the metric field is a "background" field evolving in space and time but independently from the moving mesh. To know the value of the metric at a vertex, an interpolation has to be performed. But, the representation of this metric on a background mesh is problematic due to the displacement of the boundaries and to the definition of an adequate background mesh to support the metric field. That is why the approach considered in this work is a Lagrangian approach: a metric is attached to a moving vertex and moves with it, *i.e.,* we have $\mathcal{M}(\mathbf{x}(t),t)$ which we will write $\mathcal{M}(\mathbf{x}(t))$ in what follows.

In the context of dynamic meshes, the optimization problem is still given by Relations (2) and (3) where the space-time error model is given by Relation (4). As in Section 2.3, to perform the space-time L^p error analysis with time sub-intervals, we first do a spatial minimization for a given sub-interval. We consider the i^{th} sub-interval $[t^i, t^{i+1}]$. Given the continuous mesh spatial complexity $\mathcal{N}^i$, we seek for the optimal Arbitrary-Lagrangian-Eulerian (ALE) continuous mesh $\mathbf{M}^{i,\mathrm{ALE}}_{L^p} = \left(\mathcal{M}^{i,\mathrm{ALE}}_{L^p}(\mathbf{x}(t))\right)_{\mathbf{x}\in\Omega(t)}$ for the whole sub-interval which is solution of the following problem:

$$\mathbf{E}^{i,\mathrm{ALE}}_{L^p}(\mathbf{M}^{i,\mathrm{ALE}}_{L^p}) = \min_{\mathbf{M}^i} \int_{t^i}^{t^{i+1}} \left(\int_{\Omega(t)} \mathrm{trace}\left((\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}(t)) \, |H_u(\mathbf{x}(t),t)| \, (\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}(t)) \right)^p \mathrm{d}\mathbf{x}(t) \right) \mathrm{d}t \quad \text{such that} \quad C(\mathbf{M}^i) = \mathcal{N}^i \, .$$

Note that the continuous mesh spatial complexity is constant on this sub-interval. We name it optimal ALE continuous mesh to stress that the continuous mesh is dynamic (*i.e.,* it evolves in time), it represents a moving adapted mesh and it is coupled with an ALE flow solver. Now $\mathbf{M}^i$ has a dependence in time, thus we cannot move the integral over time into the trace to make the mean Hessian metric appear like in the fixed mesh analysis. The spatial minimization for a given sub-interval cannot be done directly.

To solve this issue, we first exhibit the optimal instantaneous ALE continuous mesh minimizing the interpolation error in L^p norm at a given time t, also shortly called ALE metric field, that takes into account the mesh deformation (Section 4.1) and then we validate it on several analytical examples (Section 4.2). This ALE metric field makes it possible to map a continuous moving mesh of $\Omega(t)$, $t \in [t^i, t^{i+1}]$, onto a continuous mesh of $\Omega(t^i)$. Then, the error model can be written only on $\Omega(t^i)$ and the spacial minimization can be performed (Section 4.3). In other words, to preserve the adaptation of the mesh despite the mesh deformation, we are looking for a mesh at the beginning of each sub-interval that will be adapted to the solution at each time-step of the sub-interval once moved with the prescribed mesh displacement.

4.1. Optimal instantaneous ALE continuous mesh minimizing the interpolation error in L^p norm: the ALE metric

Before dealing with the case of the mesh adaptation for a sub-interval, let us solve the previous mentioned problem for one time step. The computational domain is time-dependent $\Omega(t) \subset \mathbb{R}^3 \times [0,T]$. The problem for one time-step is the following. t^n and t^{n+1} are two times, Ω^n and Ω^{n+1} are the spatial domains at t^n and t^{n+1} respectively, and generally we have $\Omega^n \neq \Omega^{n+1}$. We denote by $\mathbf{d}$ the given mesh displacement field. Then, we want to find the optimal continuous mesh $\mathbf{M}^{n,\mathrm{ALE}}_{L^p} = (\mathcal{M}^{n,\mathrm{ALE}}_{L^p}(\mathbf{x}))_{\mathbf{x}\in\Omega^n}$ defined on Ω^n from which we will generate a mesh at time t^n that, once moved with displacement $\mathbf{d}$, will be adapted to a sensor u^{n+1} at time t^{n+1} on domain Ω^{n+1}. We consider that u^{n+1} is a scalar sensor function, the extension to vector sensor functions being straightforward. The resolution of this problem leads to the optimal instantaneous ALE continuous mesh defined on Ω^n minimizing the interpolation error of sensor u^{n+1} in L^p norm at time t^{n+1} on Ω^{n+1} after being deformed by displacement $\mathbf{d}$. This ALE metric field involves the gradient of the mesh transformation between t^n and t^{n+1} to take into account the mesh deformation. A first draft of this analysis was given in [7].

For the coming mathematical analysis involving spatial derivatives, it is very important to state on which domain/mesh these derivatives are computed. To this end, the following notations will be used:

- ∇^n denotes the gradient operator[3] performed on domain Ω^n, *i.e.,* computed on mesh $\mathcal{H}^n$
- H^{n+1} denotes the Hessian operator performed on domain Ω^{n+1}, *i.e.,* computed on mesh $\mathcal{H}^{n+1}$. As operator H^{n+1} is always applied to sensor u^{n+1} at time t^{n+1}, simplified notation H_u^{n+1} will stand for $H^{n+1}[u^{n+1}]$
- $\mathcal{M}_{L^p}^{n+1}[u^{n+1}]$ denotes the point-wise optimal L^p metric for sensor u^{n+1} computed on domain Ω^{n+1}, *i.e.,* on mesh $\mathcal{H}^{n+1}$. Again, simplified notation $\mathcal{M}_{L^p}^{n+1}$ will stand for $\mathcal{M}_{L^p}^{n+1}[u^{n+1}]$
- $\mathbf{M}_{L^p}^{n+1} = (\mathcal{M}_{L^p}^{n+1}(\mathbf{x}))_{\mathbf{x}\in\Omega^{n+1}}$ is the associated continuous mesh with complexity $C(\mathbf{M}_{L^p}^{n+1}) = \mathcal{N}^{n+1}$.

Now, let us introduce $\boldsymbol{\phi}$ the mapping between domains[4] Ω^n and Ω^{n+1}:

$$\begin{array}{rccl} \boldsymbol{\phi}: & \Omega^n & \longrightarrow & \Omega^{n+1} \\ & \mathbf{x}^n & \longmapsto & \mathbf{x}^{n+1} = \boldsymbol{\phi}\left(\mathbf{x}^n\right), \end{array} \tag{16}$$

and $\mathbf{d}$ the corresponding mesh displacement field, such that:

$$\mathbf{x}^{n+1} = \boldsymbol{\phi}(\mathbf{x}^n) = \mathbf{x}^n + \mathbf{d}(\mathbf{x}^n). \tag{17}$$

Since $\boldsymbol{\phi}$ is a diffeomorphism, we have, for any infinitesimal vector $\delta\mathbf{x}^n \in \Omega^n$:

$$\delta\mathbf{x}^{n+1} = \left[\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\right]^T \delta\mathbf{x}^n \qquad \text{with} \qquad \nabla^n\boldsymbol{\phi}(\mathbf{x}^n) = \mathcal{I} + \nabla^n\mathbf{d}(\mathbf{x}^n), \quad \forall\mathbf{x}^n \in \Omega^n. \tag{18}$$

Finally, a $\widehat{.}$ operator is defined which transports a quantity from Ω^{n+1} to Ω^n. We note $\widehat{H_u^{n+1}}$ the Hessian of u^{n+1} computed on Ω^{n+1} and transported on domain Ω^n. This mathematically writes:

$$\begin{array}{rccl} \widehat{H_u^{n+1}}: & \Omega^n & \longrightarrow & \mathbb{R} \\ & \mathbf{x}^n & \longmapsto & H_u^{n+1}(\boldsymbol{\phi}(\mathbf{x}^n)) = \widehat{H_u^{n+1}}(\mathbf{x}^n). \end{array}$$

According to error analysis of Section 2.2, the optimal instantaneous continuous mesh $\mathbf{M}_{L^p}^{n+1} = (\mathcal{M}_{L^p}^{n+1}(\mathbf{x}))_{\mathbf{x}\in\Omega^{n+1}}$ at time t^{n+1} is given by Relation (5). Thus, an optimal mesh of Ω^{n+1} adapted to u^{n+1} can be built by generating a unit mesh $\mathcal{H}^{n+1}$ with respect to $\mathbf{M}_{L^p}^{n+1}$. This means that the length of any arbitrary edge $\mathbf{e}^{n+1}$ of $\mathcal{H}^{n+1}$ is one in metric $\mathcal{M}_{L^p}^{n+1}$ (see Section 2.1):

$$1 = (\mathbf{e}^{n+1}(\mathbf{x}^{n+1}))^T \ \mathcal{M}_{L^p}^{n+1}(\mathbf{x}^{n+1}) \ \mathbf{e}^{n+1}(\mathbf{x}^{n+1}), \quad \forall\mathbf{e}^{n+1}(\mathbf{x}^{n+1}) \in \mathcal{H}^{n+1}. \tag{19}$$

Let $\mathbf{e}^n$ be the edge of $\mathcal{H}^n$ having $\mathbf{e}^{n+1}$ as image by $\boldsymbol{\phi}$ in $\mathcal{H}^{n+1}$. As we are only interested in controlling the prevailing term of the interpolation error, we can use the following relation:

$$\mathbf{e}^{n+1}(\mathbf{x}^{n+1}) = \mathbf{e}^{n+1}\left(\boldsymbol{\phi}(\mathbf{x}^n)\right) = \left[\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\right]^T \mathbf{e}^n(\mathbf{x}^n), \tag{20}$$

which is true at first order in the demonstration. The idea is to unravel how Condition (19) writes when transposed onto mesh $\mathcal{H}^n$. Using Relations (5) and (20), unit length Equation (19) can be re-written:

$$\begin{aligned} 1 &= \left(\mathbf{e}^{n+1}\left(\boldsymbol{\phi}(\mathbf{x}^n)\right)\right)^T \ \mathcal{M}_{L^p}^{n+1}\left(\boldsymbol{\phi}(\mathbf{x}^n)\right) \ \mathbf{e}^{n+1}\left(\boldsymbol{\phi}(\mathbf{x}^n)\right) \\ &= \left(\left[\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\right]^T \mathbf{e}^n(\mathbf{x}^n)\right)^T \left(\left(\frac{\mathcal{N}^{n+1}}{\mathcal{K}^{n+1}}\right)^{\frac{2}{3}} \left(\det\left|H_u^{n+1}(\boldsymbol{\phi}(\mathbf{x}^n))\right|\right)^{-\frac{1}{2p+3}} \left|H_u^{n+1}(\boldsymbol{\phi}(\mathbf{x}^n))\right|\right) \left(\left[\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\right]^T \mathbf{e}^n(\mathbf{x}^n)\right) \\ &= \left(\mathbf{e}^n(\mathbf{x}^n)\right)^T \left(\left(\frac{\mathcal{N}^{n+1}}{\mathcal{K}^{n+1}}\right)^{\frac{2}{3}} \left(\det\left|\widehat{H_u^{n+1}}(\mathbf{x}^n)\right|\right)^{-\frac{1}{2p+3}} \nabla^n\boldsymbol{\phi}(\mathbf{x}^n) \left|\widehat{H_u^{n+1}}(\mathbf{x}^n)\right| \left[\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\right]^T\right) \mathbf{e}^n(\mathbf{x}^n) \\ &= \left(\mathbf{e}^n(\mathbf{x}^n)\right)^T \mathcal{M}_{L^p}^{n,\mathrm{ALE}}(\mathbf{x}^n) \ \mathbf{e}^n(\mathbf{x}^n). \end{aligned}$$

Consequently, if we create a unit mesh of Ω^n with respect to ALE continuous mesh $\mathbf{M}_{L^p}^{n,\mathrm{ALE}} = \left(\mathcal{M}_{L^p}^{n,\mathrm{ALE}}(\mathbf{x})\right)_{\mathbf{x}\in\Omega^n}$, rewriting the above calculus upside down, we get the following implication:

$$\left(\mathbf{e}^n(\mathbf{x}^n)\right)^T \mathcal{M}_{L^p}^{n,\mathrm{ALE}}(\mathbf{x}^n) \ \mathbf{e}^n(\mathbf{x}^n) = 1 \implies \left(\mathbf{e}^{n+1}(\mathbf{x}^{n+1})\right)^T \mathcal{M}_{L^p}^{n+1}(\mathbf{x}^{n+1}) \ \mathbf{e}^{n+1}(\mathbf{x}^{n+1}) = 1,$$

[3] Here, the gradient is not the Jacobian, *i.e.,* for an arbitrary vector field $\mathbf{f} = (f_1, .., f_k)$, its gradient matrix is $\nabla\mathbf{f} = \left(\frac{\partial f_j}{\partial x_i}\right)_{ij}$.

[4] Even if $\Omega^n \neq \Omega^{n+1}$ in general, we assume there is diffeomorphism mapping $\boldsymbol{\phi}$ from Ω^n to Ω^{n+1}, *.i.e.,* these two spatial domains can be mapped one onto the other.

for all edges $\mathbf{e}^n$ of mesh $\mathcal{H}^n$ having $\mathbf{e}^{n+1}$ as image by $\boldsymbol{\phi}$ in mesh $\mathcal{H}^{n+1}$. Therefore, if a unit mesh is generated with respect to $\mathbf{M}_{L^p}^{n,\mathrm{ALE}}$ at t^n, then the deformed mesh moved with displacement $\mathbf{d}$ until time t^{n+1} is unit for the optimal metric associated with sensor u^{n+1}, meaning that it is optimal to control the interpolation error in L^p norm of the sensor u at t^{n+1}.

The optimal instantaneous ALE continuous mesh reads:

$$\mathcal{M}_{L^p}^{n,\mathrm{ALE}}(\mathbf{x}^n) = \left(\frac{\mathcal{N}^{n+1}}{\mathcal{K}^{n+1}}\right)^{\frac{2}{3}} \left(\det |\widehat{H_u^{n+1}}(\mathbf{x}^n)|\right)^{-\frac{1}{2p+3}} \nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\, \big|\widehat{H_u^{n+1}}(\mathbf{x}^n)\big|\big[\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\big]^T,$$

with $\mathcal{K}^{n+1} = \displaystyle\int_{\Omega^{n+1}} \left(\det |H_u^{n+1}(\mathbf{x}^{n+1})|\right)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}^{n+1}$. The number of vertices of the mesh remains the same when it is moved with displacement $\mathbf{d}$. Indeed, for our moving mesh simulations, connectivity-changes can occur but no vertex addition or deletion is allowed (see Section 5.1). Consequently, the continuous mesh complexity remains constant in time:

$$C\left(\mathbf{M}_{L^p}^{n+1}\right) = C\left(\mathbf{M}_{L^p}^{n,\mathrm{ALE}}\right) \iff \int_{\Omega^{n+1}} \sqrt{\det \mathcal{M}_{L^p}^{n+1}(\mathbf{x}^{n+1})}\mathrm{d}\mathbf{x}^{n+1} = \int_{\Omega^n} \sqrt{\det \mathcal{M}_{L^p}^{n,\mathrm{ALE}}(\mathbf{x}^n)}\mathrm{d}\mathbf{x}^n,$$

thus, according to the definition of $\mathcal{K}$ and using $\det(\alpha\mathcal{A}) = \alpha^3 \det\mathcal{A}$ (we are in $\mathbb{R}^3$), we have:

$$\begin{aligned}\mathcal{K}^{n+1} = \mathcal{K}^{n,\mathrm{ALE}} &= \int_{\Omega^n} \left(\det\Big((\det |\widehat{H_u^{n+1}}(\mathbf{x}^n)|)^{-\frac{1}{2p+3}}\, \nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\, \big|\widehat{H_u^{n+1}}(\mathbf{x}^n)\big|\big[\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\big]^T\Big)\right)^{\frac{1}{2}} \mathrm{d}\mathbf{x}^n \\ &= \int_{\Omega^n} \big|\det\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\big| \left(\det |\widehat{H_u^{n+1}}(\mathbf{x}^n)|\right)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}^n\end{aligned}$$

Moreover, $\mathcal{M}_{L^p}^{n,\mathrm{ALE}}$ can be re-written to recover the same form as Relation (5) which will be useful for the error analysis involving dynamic meshes. To this end, we set:

$$|H_u^{n,\mathrm{ALE}}(\mathbf{x}^n)| = \big|\det\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\big|^{\frac{1}{p}} \left(\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\, |\widehat{H_u^{n+1}}(\mathbf{x}^n)|\, \big[\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\big]^T\right), \tag{21}$$

from which we deduce:

$$\det |H_u^{n,\mathrm{ALE}}(\mathbf{x}^n)| = \big|\det\nabla^n\boldsymbol{\phi}(\mathbf{x}^n)\big|^{\frac{3}{p}+2} \det |\widehat{H_u^{n+1}}(\mathbf{x}^n)| \quad \text{and} \quad \mathcal{K}^{n,\mathrm{ALE}} = \int_{\Omega^n} \left(\det |H_u^{n,\mathrm{ALE}}(\mathbf{x}^n)|\right)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}^n.$$

After some algebra, we finally get:

$$\mathcal{M}_{L^p}^{n,\mathrm{ALE}}(\mathbf{x}^n) = \left(\frac{\mathcal{N}^{n+1}}{\mathcal{K}^{n,\mathrm{ALE}}}\right)^{\frac{2}{3}} \left(\det |H_u^{n,\mathrm{ALE}}(\mathbf{x}^n)|\right)^{-\frac{1}{2p+3}} |H_u^{n,\mathrm{ALE}}(\mathbf{x}^n)|.$$

According to Section 2.2, this expression of the ALE metric field represents the optimal instantaneous continuous mesh minimizing the interpolation error in L^p norm for the sensor having as second derivatives $H_u^{n,\mathrm{ALE}}$ on Ω^n. Therefore, we can derive the expression of the interpolation error associated with continuous mesh $\mathbf{M}_{L^p}^{n,\mathrm{ALE}}$. The instantaneous interpolation error in L^p norm at the power p is given by Relation (6):

$$\begin{aligned}\widetilde{\mathbf{E}}_{L^p}(\mathbf{M}_{L^p}^{n,\mathrm{ALE}}) &= \int_{\Omega^n} \mathrm{trace}\left((\mathbf{M}_{L^p}^{n,\mathrm{ALE}})^{-\frac{1}{2}}(\mathbf{x}^n)\, |H_u^{n,\mathrm{ALE}}(\mathbf{x}^n)|\, (\mathbf{M}_{L^p}^{n,\mathrm{ALE}})^{-\frac{1}{2}}(\mathbf{x}^n)\right)^p \mathrm{d}\mathbf{x}^n \\ &= 3^p\, (\mathcal{N}^{n+1})^{\frac{2p}{3}}\, (\mathcal{K}^{n,\mathrm{ALE}})^{\frac{2p+3}{3}} = 3^p\, (\mathcal{N}^{n+1})^{\frac{2p}{3}}\, (\mathcal{K}^{n+1})^{\frac{2p+3}{3}} \\ &= \int_{\Omega^{n+1}} \mathrm{trace}\left((\mathbf{M}_{L^p}^{n+1})^{-\frac{1}{2}}(\mathbf{x}^{n+1})\, |H_u^{n+1}(\mathbf{x}^{n+1})|\, (\mathbf{M}_{L^p}^{n+1})^{-\frac{1}{2}}(\mathbf{x}^{n+1})\right)^p \mathrm{d}\mathbf{x}^{n+1} = \widetilde{\mathbf{E}}_{L^p}(\mathbf{M}_{L^p}^{n+1}).\end{aligned}$$

Now, we can state the main result:

Proposition 1 (Optimal instantaneous ALE continuous mesh)**.** *Let Ω^n and Ω^{n+1} be two spatial domains at t^n and t^{n+1}, and $\boldsymbol{\phi}$ the mapping between these domains. The optimal instantaneous ALE continuous mesh $\mathbf{M}_{L^p}^{n,\mathrm{ALE}} = \left(\mathcal{M}_{L^p}^{n,\mathrm{ALE}}(\mathbf{x})\right)_{\mathbf{x}\in\Omega^n}$ defined on Ω^n reads:*

$$\mathcal{M}_{L^p}^{n,\mathrm{ALE}}(\mathbf{x}) = \left(\mathcal{N}^{n+1}\right)^{\frac{2}{3}} \left(\int_{\Omega^n} \left(\det |H_u^{n,\mathrm{ALE}}(\bar{\mathbf{x}})|\right)^{\frac{p}{2p+3}} \mathrm{d}\bar{\mathbf{x}}\right)^{-\frac{2}{3}} \left(\det |H_u^{n,\mathrm{ALE}}(\mathbf{x})|\right)^{-\frac{1}{2p+3}} |H_u^{n,\mathrm{ALE}}(\mathbf{x})|, \tag{22}$$

with the ALE Hessian-metric given by

$$|H_u^{n,\mathrm{ALE}}(\mathbf{x})| = \big|\det\nabla^n\boldsymbol{\phi}(\mathbf{x})\big|^{\frac{1}{p}} \left(\nabla^n\boldsymbol{\phi}(\mathbf{x})\, |\widehat{H_u^{n+1}}(\mathbf{x})|\, \big[\nabla^n\boldsymbol{\phi}(\mathbf{x})\big]^T\right).$$

The following properties hold:

i) *Let us assume continuous mesh $\mathbf{M}^{n,\mathrm{ALE}}_{L^p}$ is used to generate a unit mesh $\mathcal{H}^n_{\mathrm{ALE}}$ of Ω^n and let us denote by $\mathcal{H}^{n+1}_{\mathrm{ALE}}$ the mesh of Ω^{n+1} which is the image of mesh $\mathcal{H}^n_{\mathrm{ALE}}$ by mapping $\boldsymbol{\phi}$. Then, mesh $\mathcal{H}^{n+1}_{\mathrm{ALE}}$ is optimal to control the interpolation error in L^p norm of sensor u^{n+1} on Ω^{n+1}.*

ii) *Continuous mesh $\mathbf{M}^{n,\mathrm{ALE}}_{L^p}$ has the same complexity $\mathcal{N}^{n+1}$ as continuous mesh $\mathbf{M}^{n+1}_{L^p}$.*

iii) *Continuous mesh $\mathbf{M}^{n,\mathrm{ALE}}_{L^p}$ achieves on Ω^n the same level of interpolation error as continuous mesh $\mathbf{M}^{n+1}_{L^p}$ on Ω^{n+1}:*

$$\widetilde{\mathbf{E}}_{L^p}(\mathbf{M}^{n,\mathrm{ALE}}_{L^p}) = \widetilde{\mathbf{E}}_{L^p}(\mathbf{M}^{n+1}_{L^p}) \ \text{ where } \widetilde{\mathbf{E}}_{L^p} \text{ is given by Relation (6).}$$

iv) *There is no reason for adapted mesh $\mathcal{H}^n_{\mathrm{ALE}}$, which is a unit mesh for $\mathbf{M}^{n,\mathrm{ALE}}_{L^p}$, to be optimal for the control of the interpolation error of sensor u^n at t^n.*

4.2. Validation on two-dimensional analytic examples

We validate the result of Proposition 1 on analytic examples in two dimensions, three-dimensional examples being available in [9, 44]. A uniform mesh $\mathcal{H}^n_0$ of domain $\Omega = [-1, 1]^2$, a displacement field $\mathbf{d}$ between two times t^n and t^{n+1}, and a sensor u^{n+1} at time t^{n+1} are given. We choose to control the interpolation error in L^1 norm and theoretical spatial complexities from $10,000$ to $400,000$ are prescribed to perform a convergence study. We generate a unit mesh $\mathcal{H}^{n,\mathrm{ALE}}$ with respect to $\mathbf{M}^{n,\mathrm{ALE}}_{L^1}$ (given by Relation (22)) at time t^n that, once moved into $\mathcal{H}^{n+1,\mathrm{ALE}} = \boldsymbol{\phi}(\mathcal{H}^{n,\mathrm{ALE}})$ is expected to be adapted to the sensor u^{n+1} according to the above developments. At the same time, a reference adapted mesh $\mathcal{H}^{n+1}$ is generated that is directly adapted to sensor u^{n+1} using optimal continuous mesh $\mathbf{M}^{n+1}_{L^1}$ (given by Relation (5)). To validate the optimality of $\mathcal{H}^{n+1,\mathrm{ALE}}$, we are going to compare the interpolation error in L^1 norm of sensor u^{n+1} on both meshes and analyse the quality of the meshes elements. Algorithm 2 describes the procedure to generate both meshes. Note that, for the ALE cases, the meshes are directly moved from their initial position at t^n to their final position at t^{n+1} in one step. No mesh optimizations are performed after the displacement.

Algorithm 2 Procedure to generate adapted meshes with the ALE L^1 metric field and the regular L^1 metric field.

- Generation of an adapted mesh with the optimal ALE L^1 metric field
  1. $\nabla^n\boldsymbol{\phi}$ = Compute transformation gradient $\left(\mathcal{H}^n_0, \mathbf{d}\right)$
  2. $\mathcal{H}^{n+1}_0$ = Move mesh $\left(\mathcal{H}^n_0, \mathbf{d}\right)$
  3. u^{n+1} = Compute target sensor $\left(\mathcal{H}^{n+1}_0\right)$
  4. $\mathbf{M}^{n,\mathrm{ALE}}_{L^1}$ = Compute ALE metric field $\left(\mathcal{H}^{n+1}_0, u^{n+1}, \mathcal{H}^n_0, \nabla^n\boldsymbol{\phi}\right)$
  5. $\mathcal{H}^{n,\mathrm{ALE}}$ = Adapt mesh $\left(\mathcal{H}^n_0, \mathbf{M}^{n,\mathrm{ALE}}_{L^1}\right)$
  6. $\mathcal{H}^{n+1,\mathrm{ALE}}$ = Move mesh $\left(\mathcal{H}^{n,\mathrm{ALE}}, \mathbf{d}\right)$
- Generation of an adapted mesh with the optimal L^1 metric field
  1. u^{n+1} = Compute target sensor $\left(\mathcal{H}^n_0\right)$
  2. $\mathbf{M}^{n+1}_{L^1}$ = Compute metric field $\left(\mathcal{H}^n_0, u^{n+1}\right)$
  3. $\mathcal{H}^{n+1}$ = Adapt mesh $\left(\mathcal{H}^n_0, \mathbf{M}^{n+1}_{L^1}\right)$

The following two-dimensional analytic functions are used as sensor functions:

$$u^{n+1}_1(x,y) = \begin{cases} 0.01\,\sin(50xy) & \text{if } |xy| \geq \dfrac{2\pi}{50} \\ \sin(50xy) & \text{if } |xy| < \dfrac{2\pi}{50} \end{cases} \quad \text{and} \quad u^{n+1}_2(x,y) = 0.1\,\sin(50\,x) + \arctan\left(\frac{0.1}{\sin(5\,y) - 2\,x}\right).$$

They are shown in Figures 5 and 6. First sensor exhibits features of different scales, small oscillations of amplitude 0.01 and large oscillations of amplitude 1, which makes it a suitable example for our multiscale mesh adaptation process. The second sensor exhibits a strong discontinuity-like sinusoidal feature - through the arctangent - crossing small amplitude waves.

Two displacements $\mathbf{d}_i : \Omega^n \longrightarrow \Omega^{n+1}$ are considered:

$$\mathbf{d}_1(x,y) = \begin{bmatrix} \begin{cases} -0.3\,(x+1)\left(y^2-1\right)\exp\left(-5x^2\right), & \text{if } x \geq 0 \\ 0.3\,(x-1)\left(y^2-1\right)\exp\left(-5x^2\right), & \text{if } x < 0 \end{cases} \\ \\ \begin{cases} -0.3\left(x^2-1\right)(y+1)\exp\left(-5y^2\right), & \text{if } y \geq 0 \\ 0.3\left(x^2-1\right)(y-1)\exp\left(-5y^2\right), & \text{if } y < 0 \end{cases} \end{bmatrix} \quad \text{and} \quad \mathbf{d}_2(x,y) = \begin{bmatrix} 0.5\,(x^2-1)\,(y^2-1) \\ 0 \end{bmatrix}.$$

They are illustrated in Figure 4 where the displacements have been applied to a uniform-size mesh composed of 53, 262 vertices

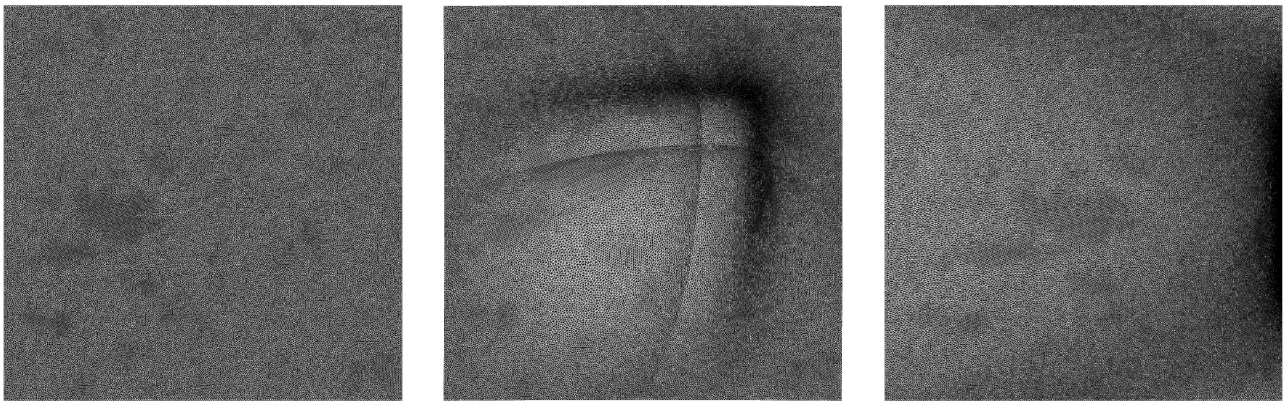

Figure 4: Left, an initial uniform-size mesh. Middle and right, resulting meshes after applying displacements $\mathbf{d}_1^{n+1}$ (midlle) and $\mathbf{d}_2^{n+1}$ (right) to the initial uniform-size mesh. These displacements are used to validate the ALE metric formulation.

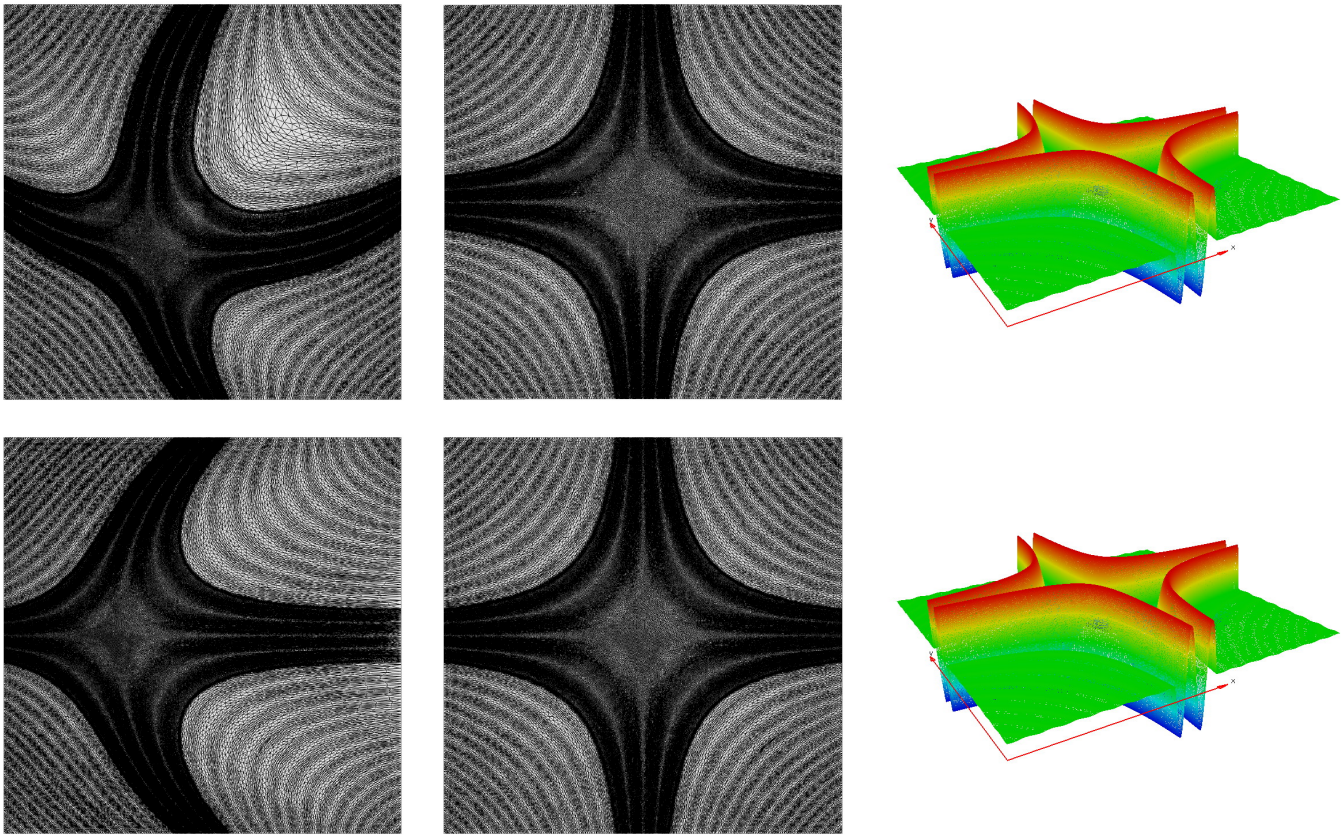

Figure 5: Mesh adapted with the ALE metric for a spatial complexity of 50, 000 at time t^n (left) and t^{n+1} (middle) - *i.e.,* after moving the mesh - for analytical function u_1^{n+1} (right), and displacements $\mathbf{d}_1$ (top) and $\mathbf{d}_2$ (bottom).

to emphasize the mesh distortion induced by these mesh deformations. It is important to note that these functions lead to very large displacement: $\|\mathbf{d}_1\|_{\max} \approx 0.1944$ and $\|\mathbf{d}_2\|_{\max} = 0.5$ which are a lot larger than the mesh size in all cases. Hence, even if the theory considers infinitesimal displacement, these test cases validate the result in a more realistic context.

Figures 5 and 6 show the resulting meshes for both sensors and both displacements for a prescribed theoretical spatial complexity of $50,000$ leading to discrete adapted meshes composed of almost $60,000$ vertices. Initial adapted meshes $\mathcal{H}^{n,\mathrm{ALE}}$ at time t^n are displayed on the left picture, and the resulting adapted mesh $\mathcal{H}^{n+1,\mathrm{ALE}}$ at time t^{n+1} after applying the mesh displacement are shown on the middle picture. Note that the optimal meshes $\mathcal{H}^{n+1}$ obtained directly from $\mathbf{M}^{n+1}_{L^1}$ are very similar to the meshes shown on the middle picture.

A quantitative analysis of these examples is required to make sure our ALE procedure results in effectively adapted meshes. To this end, we compare the mesh generated with a classic adaptation procedure and the mesh generated with the ALE procedure through two quantities:

- The interpolation error $\|u^{n+1} - \Pi_h u^{n+1}\|_\Omega$ committed when the sensor function is projected on the mesh remains the best way to evaluate if the mesh is really adapted to the considered sensor.
- To analyse any possible distortion of the mesh due to the mesh deformation, we can check that the shape of the elements is in accordance with the reference adapted metric. To this end, we compute the mesh elements quality with respect to metric field $\mathbf{M}^{n+1}_{L^1}$ used to generate the reference adapted mesh. The 2D element quality in the metric is given by:

$$Q_{\mathcal{M}^{n+1}_{L^1}}(K) = \frac{\sqrt{3}}{12} \frac{\sum_{i=1}^{3} \ell^2_{\mathcal{M}^{n+1}_{L^1}}(\mathbf{e}_i)}{|K|_{\mathcal{M}^{n+1}_{L^1}}} \quad \in \quad [1, +\infty]$$

where the $\mathbf{e}_i$ are the three edges of the triangle, and length and area formula in the metric are given in Section 2.1. To give an idea, $Q_{\mathcal{M}}(K) = 1$ corresponds to a perfectly regular element in the metric, $Q_{\mathcal{M}}(K) < 2$ correspond to excellent quality elements, while a high value (> 100) of $Q_{\mathcal{M}}(K)$ indicates a nearly degenerate element.

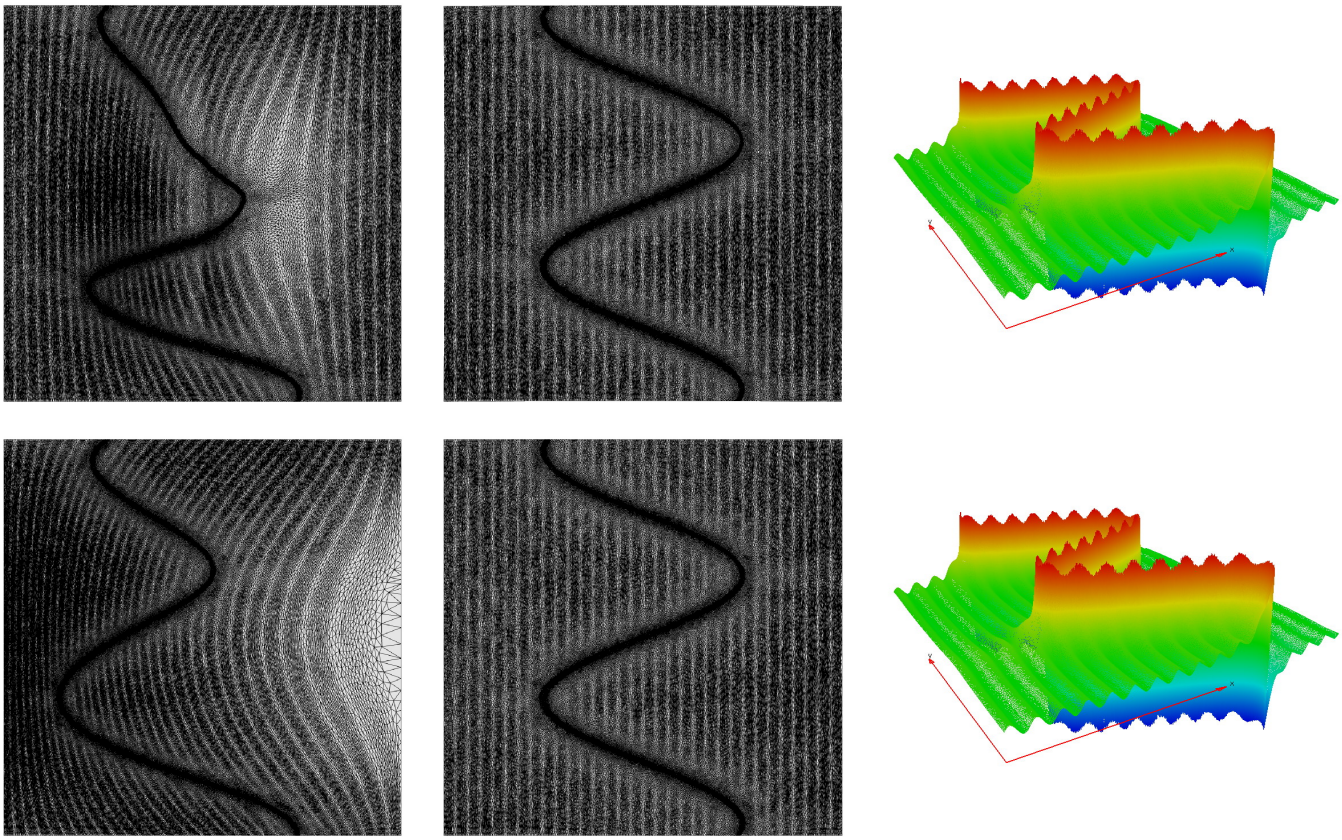

Figure 6: Mesh adapted with the ALE metric for a spatial complexity of $50,000$ at time t^n (left) and t^{n+1} (middle) - *i.e.,* after moving the mesh - for analytical function u_2^{n+1} (right), and displacements $\mathbf{d}_1$ (top) and $\mathbf{d}_2$ (bottom).

The interpolation error convergence curves are represented in Figure 7 where the interpolation error in L^1 norm is plotted with respect to the number of vertices. We observe that, for both sensors and both displacements, the interpolation error obtained with the ALE continuous mesh is almost similar or only slightly higher than the interpolation error obtained with the direct adaptation. This slight difference is due notably to approximations made when establishing the formulation of the ALE continuous mesh whereas large displacement are performed between t^n and t^{n+1}. This confirms the optimality of the ALE continuous mesh, *i.e.,* the property *i*) of Proposition 1. Note that the interpolation error is one order of magnitude smaller on adapted meshes than on uniform meshes, or that ten times more vertices are required on uniform meshes to achieve the same level of interpolation error. This clearly establishes the interest of the adaptation.

In regard to mesh elements quality, results are given in Tables 1 and 2. The reference adapted meshes have an almost perfect quality with respect to the prescribed metric $\mathbf{M}_{L^1}^{n+1}$, which was expected and is achieved thanks to the efficient local remesher `Amg` [39]. The quality of the meshes adapted with the ALE procedure at time t^{n+1} with respect to metric $\mathbf{M}_{L^1}^{n+1}$ is very good, which shows that resulting adapted meshes from both standard and ALE continuous mesh are very similar in terms of element-shape quality. This points out that the mesh deformation is perfectly taken into account in the ALE metric field formulation. Again, this validates property *i*) of Proposition 1.

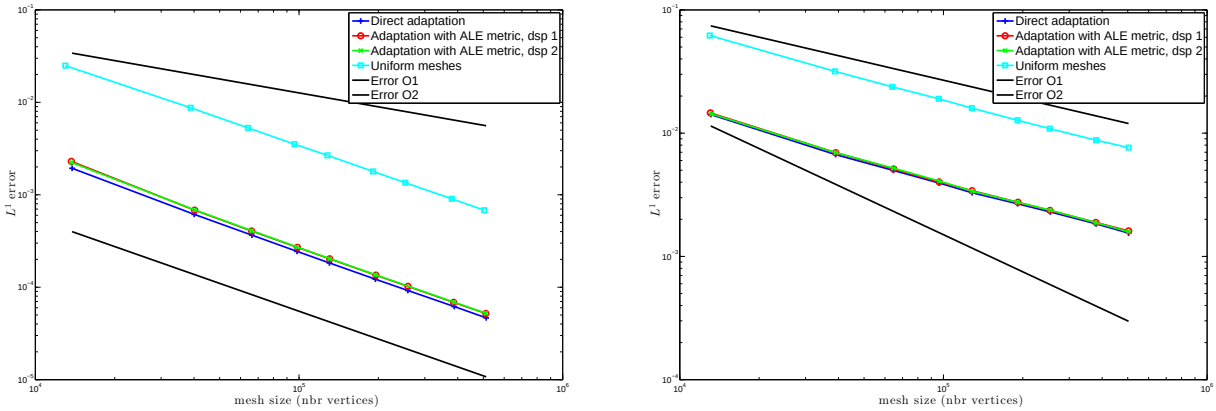

Figure 7: Interpolation error convergence curves for sensor u_1^{n+1} (left) and u_2^{n+1} (right).

	$\mathcal{H}_{ref}^{n+1}$		$\mathcal{H}_{\text{ALE}}^{n+1}$ with $\mathbf{d}_1$		$\mathcal{H}_{\text{ALE}}^{n+1}$ with $\mathbf{d}_2$	
Average quality	1.07		1.20		1.23	
Worst quality	3.28		23.7		66.0	
$1.00 < Q < 2.00$	127303	99.92 %	122473	96.69 %	121585	96.02 %
$2.00 < Q < 3.00$	97	0.08 %	3283	2.59 %	3641	2.88 %
$3.00 < Q < 4.00$	6	0.00 %	611	0.48 %	843	0.67 %
$4.00 < Q < 5.00$	0	0.00 %	170	0.13 %	244	0.19 %
$5.00 < Q < 10.00$	0	0.00 %	121	0.10 %	258	0.20 %
$10.00 < Q < 50.00$	0	0.00 %	7	0.01 %	49	0.04 %
$50.00 < Q < 100.00$	0	0.00 %	0	0.00 %	2	0.00 %

Table 1: Adapted meshes elements quality for sensor u_1^{n+1} and a spatial complexity of $50,000$.

4.3. Space-time error analysis for dynamic meshes

Section 4.1 provides the optimal instantaneous ALE continuous mesh which takes into account the mesh deformation. Now, we can extend the space-time error analysis with time sub-intervals done for fixed meshes - Section 2.3 - to the case of dynamic meshes. The simulation time interval is split into n_{adap} sub-intervals. On each sub-interval, the mesh size (number of vertices) remains constant, but the mesh is deformed to follow the geometry displacement. At each time-step of the sub-interval, we want the moved mesh to be adapted to the current sensor. The key idea to perform the error analysis is to seek for the optimal dynamic continuous mesh at the beginning of the sub-interval, this continuous mesh being optimal for the whole sub-interval when deformed, instead of seeking for the expression of the optimal continuous mesh at each instant, *i.e.,* as a function of the time.

	$\mathcal{H}^{n+1}_{ref}$		$\mathcal{H}^{n+1}_{\text{ALE}}$ with $\mathbf{d}_1$		$\mathcal{H}^{n+1}_{\text{ALE}}$ with $\mathbf{d}_2$	
Average quality	1.06		1.19		1.21	
Worst quality	13.8		20.5		24.4	
$1.00 < Q < 2.00$	126406	99.95 %	121522	96.87 %	121146	96.36 %
$2.00 < Q < 3.00$	59	0.05 %	3054	2.43 %	3631	2.89 %
$3.00 < Q < 4.00$	5	0.00 %	594	0.47 %	652	0.52 %
$4.00 < Q < 5.00$	0	0.00 %	152	0.12 %	177	0.14 %
$5.00 < Q < 10.00$	0	0.00 %	118	0.09 %	109	0.09 %
$10.00 < Q < 50.00$	1	0.00 %	4	0.00 %	5	0.00 %

Table 2: Adapted meshes elements quality for sensor u_2^{n+1} and a spatial complexity of $50,000$.

To perform the spatial minimization for a sub-interval, we consider the i^{th} sub-interval $[t^i, t^{i+1}]$. Given the continuous mesh spatial complexity $\mathcal{N}^i$, we seek for the optimal ALE continuous mesh $\mathbf{M}^{i,\text{ALE}}_{L^p} = \left(\mathcal{M}^{i,\text{ALE}}_{L^p}(\mathbf{x}(t))\right)_{\mathbf{x}\in\Omega(t)}$ for the whole sub-interval which is solution of the following problem:

$$\mathbf{E}^{i,\text{ALE}}_{L^p}(\mathbf{M}^{i,\text{ALE}}_{L^p}) = \min_{\mathbf{M}^i} \int_{t^i}^{t^{i+1}} \left(\int_{\Omega(t)} \text{trace}\left((\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}(t)) \, |H_u(\mathbf{x}(t),t)| \, (\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}(t)) \right)^p \, \text{d}\mathbf{x}(t) \right) \text{d}t\,, \quad \text{such that} \quad C(\mathbf{M}^i) = \mathcal{N}^i\,. \tag{23}$$

The continuous mesh spatial complexity is constant on this sub-interval. To remove the time dependency of continuous mesh $\mathbf{M}^i$, we propose to use the optimal instantaneous ALE continuous mesh and property *iii*) of Proposition 1 which provides equality between interpolation errors. Indeed, continuous mesh $\left(\mathcal{M}^i(\mathbf{x}(t))\right)_{\mathbf{x}\in\Omega(t)}$ can be mapped back to $\Omega(t^i)$ using $\left(\mathcal{M}^{i,\text{ALE}}(\mathbf{x}(t^i))\right)_{\mathbf{x}\in\Omega(t^i)}$ where $\boldsymbol{\phi}$ maps $\Omega(t^i)$ onto $\Omega(t)$, and property *iii*) of Proposition 1 states that

$$\widetilde{\mathbf{E}}_{L^p}(\mathbf{M}^{i,\text{ALE}}) = \widetilde{\mathbf{E}}_{L^p}(\mathbf{M}^i)\,,$$

thus expression of the interpolation error for each time t can be re-written at time t^i. As we seek for the dynamic continuous mesh at time t^i which is optimal to control the interpolation error for the whole sub-interval, we can recast Error Model (23) into the following error model where the metric is independent of the time:

$$\mathbf{E}^{i,\text{ALE}}_{L^p}(\mathbf{M}^i) = \int_{t^i}^{t^{i+1}} \left(\int_{\Omega(t^i)} \text{trace}\left((\mathcal{M}^{i,\text{ALE}})^{-\frac{1}{2}}(\mathbf{x}(t^i)) \, |H^{i,\text{ALE}}_u(\mathbf{x}(t^i))| \, (\mathcal{M}^{i,\text{ALE}})^{-\frac{1}{2}}(\mathbf{x}(t^i)) \right)^p \, \text{d}\mathbf{x}(t^i) \right) \text{d}t\,,$$

where the $\mathbf{x}(t^i)$ are in domain $\Omega(t^i)$. The time dependency in $H^{i,\text{ALE}}_u$ is hidden in mapping $\boldsymbol{\phi}$ and operator $\widehat{\cdot}$. The previous expression can now be written on $\Omega(t^i)$:

$$\mathbf{E}^{i,\text{ALE}}_{L^p}(\mathbf{M}^i) = \int_{\Omega(t^i)} \text{trace}\left((\mathcal{M}^{i,\text{ALE}})^{-\frac{1}{2}}(\mathbf{x}(t^i)) \, \mathbf{H}^{i,\text{ALE}}_u(\mathbf{x}(t^i)) \, (\mathcal{M}^{i,\text{ALE}})^{-\frac{1}{2}}(\mathbf{x}(t^i)) \right)^p \, \text{d}\mathbf{x}(t^i)\,,$$

where the mean ALE Hessian metric is: $\mathbf{H}^{i,\text{ALE}}_u(\mathbf{x}(t^i)) = \displaystyle\int_{t^i}^{t^{i+1}} |H^{i,\text{ALE}}_u(\mathbf{x}(t^i))| \, \text{d}t$. The expression of the error has exactly the same form as in Problem (9) in Section 2.3, thus the spatial minimization gives the same optimal metric where $\mathbf{H}^i_u$ is replaced by $\mathbf{H}^{i,\text{ALE}}_u$. Then, the temporal minimization leads to the following optimal space-time ALE continuous mesh $\mathbf{M}^{\text{ALE}}_{L^p} = \{\mathbf{M}^{i,\text{ALE}}_{L^p}\}_{i=1,..,n_{adap}}$:

$$\mathcal{M}^{i,\text{ALE}}_{L^p}(\mathbf{x}(t^i)) = \mathcal{N}_{st}^{\frac{2}{3}} \left(\sum_{j=1}^{n_{adap}} \mathcal{K}^{j,\text{ALE}} \left(\int_{t^j}^{t^{j+1}} \tau(t)^{-1} \text{d}t \right)^{\frac{2p}{2p+3}} \right)^{-\frac{2}{3}} \left(\int_{t^i}^{t^{i+1}} \tau(t)^{-1} \text{d}t \right)^{-\frac{2}{2p+3}} (\det \mathbf{H}^{i,\text{ALE}}_u(\mathbf{x}(t^i)))^{-\frac{1}{2p+3}} \; \mathbf{H}^{i,\text{ALE}}_u(\mathbf{x}(t^i))\,,$$

where the $\mathbf{x}(t^i)$ are in domain $\Omega(t^i)$. The ALE continuous mesh dependence in time is hidden in $\mathbf{H}^{i,\text{ALE}}_u$ by means of mapping $\boldsymbol{\phi}$. This way, the mesh generated at time t^i is adapted to the solution at any time $t > t^i$ within the sub-interval $[t^i, t^{i+1}]$ once moved with the mesh deformation displacement. Once again, we stress that preserving the number of degrees of freedom when moving the mesh is essential in this analysis.

5. Global fixed-point mesh adaptation algorithm for moving meshes

The optimal space-time ALE continuous mesh is designed to fit in the global fixed-point unsteady mesh adaptation algorithm described in Algorithm 1. Nevertheless, a few things need to be modified to extend this algorithm to moving mesh ALE simulations. In this section, we first recall the connectivity-change moving mesh strategy, how connectivity-changes are handled in the ALE numerical scheme, and how the ALE metric field is updated in time. Then, we specify the modifications to Algorithm 1.

5.1. Connectivity-change moving mesh strategy

The moving mesh strategy used in this paper is detailed in [1]. The mesh adaptation framework makes us consider body-fitted simulations. Our strategy is designed to deal with large displacement of the boundaries, when the volume mesh quickly gets too distorted. The aim of this strategy is to preserve a good mesh quality all along the movement without costly remeshings, thanks to an improved mesh deformation step and an improved mesh optimization step.

The aim of the mesh deformation step is to compute a displacement for all inner vertices from the displacement of the boundaries, so that the volume mesh follows the moving geometries and thus remains valid. This is achieved by solving a linear-elasticity-like PDE on the whole domain, the inside of the meshed domain being taken for a nearly incompressible elastic material and the displacement of the boundaries being the boundary conditions of the elasticity problem. To improve the CPU efficiency of this step, the number of such solutions is significantly reduced: the mesh deformation problem is solved for large time-steps, and the trajectories are constant over these large time-steps. To improve the precision of this step, higher-order trajectories are computed: two elasticity problems are solved, thus providing a speed and an acceleration to the inner vertices. In the present strategy, boundary vertices do not slide on the surfaces (mainly because it requires an accurate geometric re-projection on a CAD model), except in the specific case of simple planar surfaces such as a symmetry plane.

To preserve a good mesh quality, local mesh optimizations are performed between two mesh deformation steps, using only vertex smoothing and generalized edge/face swapping. No vertices are added or removed. Vertex smoothing consists in moving vertices close to the center of gravity of their vertex ball, and helps recovering nicely shaped elements. The swap operator changes the connectivity of the mesh, and is especially powerful in handling shear and large deformation movement. These mesh optimizations are simulated on the current mesh considering the current metric field to evaluate quality criteria, and we also analyse their impact on the mesh at the end of the mesh deformation (*i.e.,* in the future) considering the metric field at the end of the mesh deformation. A node repositioning or an edge/face swapping is accepted and performed if and only if it satisfies the quality criteria on the current mesh and the quality criteria on the mesh at the end of the mesh deformation, see [1] for details. For instance, a face swap operation that improves the quality of the current mesh but invalidate an element at the end of the mesh deformation will be rejected. The definition of the appropriate metric field - as the metric field evolves in time - is discussed in Section 5.3. This way, the mesh optimizations cannot affect the accuracy and the directional features of the mesh, otherwise the quality check will reject such optimizations.

These optimization steps are not performed at every solver time-step, but only when vertices have crossed a certain predetermined number of elements, considering a geometric CFL-like condition (generally every few tens or hundreds of flow solver time-steps). Moreover, quality optimizations are simulated only on elements having a quality greater than a given threshold which is generally set to 3, and they are applied only if they fulfill the considered quality criteria. Therefore, most of the time, mesh optimizations impact less that one or just a few percents of the mesh elements.

Remark 1. *The connectivity-change moving mesh algorithm has been designed such that there is uniqueness of the result. If a simulation is re-run with the same data and on the same computer, then the result will be the same (the same smoothing and swapping operations will be performed). This uniqueness is also preserved in parallel.*

5.2. Dealing with connectivity-change in the ALE numerical scheme

In [10], this moving mesh algorithm has been successfully coupled to the flow solver described in Section 3.4. The numerical flow solver uses an Arbitrary Lagrangian Eulerian (ALE) numerical schemes to deal with flow simulations where the mesh is moving independently from the physical phenomena. In the ALE framework, the number of degrees of freedom and the connectivity of the mesh are supposed to be constant [12, 47]. However, the efficiency of the moving mesh algorithm relies on the connectivity-change optimization. To devise a connectivity-change moving mesh algorithm fitting within this framework, the following choices have been made. First, the number of degrees of freedom of the simulation is kept constant. In two dimensions, an ALE formulation of the swap operator was proposed in [45], but its extension to the three dimensions case is rather tedious. In [10], we have shown that the impact of swaps on the solution accuracy is very small when a linear interpolation method is considered. Nevertheless, a conservative $\mathbb{P}^1$-interpolation (see Section 3.2) is applied on each swap configuration in order to conserve the mass of the problem variables. Note that the CPU time over-cost of this conservative interpolation on each swap configuration is negligible because the number of swaps is small (it impacts only a small number of elements) at each mesh optimization and that mesh optimizations are only performed each tens or hundreds flow solver time-steps.

5.3. Update of the ALE metric field for optimizations

The mesh optimization procedure (smoothing and swapping) of the connectivity-change moving mesh strategy requires a proper metric field to evaluate geometric quantities and elements qualities of anisotropic adapted meshes. But, at a given fixed-point iteration j and sub-interval i, the input is the optimal ALE continuous mesh $\mathbf{M}_{L^p}^{i,\mathrm{ALE}}$ defined on $\Omega(t^i)$. Therefore, when the mesh is deformed during the sub-interval due to the geometry displacement, the input metric is no more compatible with the moved mesh at a given time t. The use of an incorrect metric field will drive the smoothing to move vertices to a wrong location and the swaps to break the anisotropy, thus spoiling the adaptation of the mesh. As the mesh is evolving in time, the input metric field (*i.e.,* the input continuous mesh) must also evolve in time. Consequently, we have to apply the deformation of the adapted mesh to the continuous mesh in order to maintain the consistency between the discrete and the continuous meshes.

But, in the theory developed in the previous sections, the mesh deformation correction is applied to the Hessian-metric $|H_u^{i,\mathrm{ALE}}|$ and not directly to the continuous mesh. Therefore, from the optimal ALE continuous mesh $\left(\mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}(t^i))\right)_{\mathbf{x}\in\Omega(t^i)}$ at the beginning of sub-interval $[t^i, t^{i+1}]$, we cannot directly find the deformed optimal ALE continuous mesh $\left(\mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}(t))\right)_{\mathbf{x}\in\Omega(t)}$ for $t \in [t^i, t^{i+1}]$.

Two possibilities arise from these remarks. First, several ALE metric field samples $\left\{\left(\mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}(t^k))\right)_{\mathbf{x}\in\Omega(t^k)}\right\}_k$ can be pre-calculated for each sub-interval at the previous fixed-point iteration, and when an ALE metric field is required at time t, it is linearly interpolated in time on the current mesh from two of these pre-calculated ALE metric fields. This method has two drawbacks: it requires computing and saving all these ALE metric fields and it requires a metric interpolation stage at each mesh optimization which leads to consequent memory, I/Os and CPU time overhead. The second possibility is to modify the input ALE metric field according to the current mesh deformation to get a consistant metric field at time t. To deform the continuous mesh defined at the beginning of the sub-interval, we adopt the same reasoning as the one that leads to the optimal instantaneous ALE continuous mesh in Section 4.1, but on a reverse time frame.

Let us consider sub-interval $[t^i, t^{i+1}]$, t^k a time in this sub-interval and $\mathbf{d}$ the displacement field of the mesh between t^i and t^k. Here, the problem is reversed, we seek for the continuous mesh at time t^k that will result in the continuous mesh at time t^i once moved with displacement $-\mathbf{d}$. At time t^i, the continuous mesh is $\mathbf{M}_{L^p}^{i,\mathrm{ALE}}$ (the adapted mesh at time t^i is generated directly using this metric field). Displacement $\mathbf{d}$ is the displacement of vertices between t^i and t^k: $\mathbf{x}^k = \mathbf{x}^i + \mathbf{d}(\mathbf{x}^i)$ and we write $\mathbf{d}'(\mathbf{x}^k) = \mathbf{x}^i - \mathbf{x}^k = -\mathbf{d}(\mathbf{x}^i)$. We define

$$\begin{array}{rccl} \boldsymbol{\phi}' : & \Omega^k & \longrightarrow & \Omega^i \\ & \mathbf{x}^k & \longmapsto & \mathbf{x}^i = \boldsymbol{\phi}'\left(\mathbf{x}^k\right) = \mathbf{x}^k + \mathbf{d}'(\mathbf{x}^k)\,. \end{array} \tag{24}$$

We start from an adapted mesh which is unit for metric field $\mathbf{M}_{L^p}^{i,\mathrm{ALE}}$ at time t^i and we search for the expression of the metric field at time t^k for which the deformed adapted mesh at time t^k is unit. Following Section 4.1, we consider an edge $\mathbf{e}^k$ at time t^k having edge $\mathbf{e}^i$ as image by $\boldsymbol{\phi}'$ at time t^i. They verify:

$$\begin{aligned} 1 &= \left(\mathbf{e}^i(\mathbf{x}^i)\right)^T \ \mathcal{M}_{L^p}^{i,\mathrm{ALE}}\left(\mathbf{x}^i\right) \ \left(\mathbf{e}^i(\mathbf{x}^i)\right) \\ &= \left[\mathbf{e}^i\left(\boldsymbol{\phi}'(\mathbf{x}^k)\right)\right]^T \ \mathcal{M}_{L^p}^{i,\mathrm{ALE}}\left(\mathbf{x}^i\right) \ \mathbf{e}^i\left(\boldsymbol{\phi}'(\mathbf{x}^k)\right) \\ &= \left(\left[\nabla^k\boldsymbol{\phi}'(\mathbf{x}^k)\right]^T \ \mathbf{e}^k(\mathbf{x}^k)\right)^T \ \mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}^i) \ \left(\left[\nabla^k\boldsymbol{\phi}'(\mathbf{x}^k)\right]^T \ \mathbf{e}^k(\mathbf{x}^k)\right) \\ &= \left(\mathbf{e}^k(\mathbf{x}^k)\right)^T \ \left\{\nabla^k\boldsymbol{\phi}'(\mathbf{x}^k) \ \mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}^i) \ \left[\nabla^k\boldsymbol{\phi}'(\mathbf{x}^k)\right]^T\right\} \ \left(\mathbf{e}^k(\mathbf{x}^k)\right). \end{aligned}$$

In other words, the metric field $\mathbf{M}_{L^p}^{i,optim}$ we are looking for at time t^k is:

$$\mathcal{M}_{L^p}^{i,optim}(\mathbf{x}(t^k)) = \nabla^k\boldsymbol{\phi}'\left(\mathbf{x}(t^k)\right) \ \mathcal{M}_{L^p}^{i,\mathrm{ALE}}\left(\mathbf{x}(t^i)\right) \ \left[\nabla^k\boldsymbol{\phi}'\left(\mathbf{x}(t^k)\right)\right]^T . \tag{25}$$

This metric field is easy to compute on the fly, because the central term is the input metric field $\mathcal{M}_{L^p}^{i,\mathrm{ALE}}\left(\mathbf{x}(t^i)\right)$ which is known, and the gradients of $\boldsymbol{\phi}'$ are easy to compute, since the displacement considered in $\boldsymbol{\phi}'$ is the opposite of the current displacement at each vertex. Note that the gradients of $\boldsymbol{\phi}'$ should be computed on $\Omega(t^k)$ that is to say on the current mesh.

5.4. Modifications of the global fixed-point mesh adaptation algorithm

The overall global fixed-point mesh adaptation algorithm remains unchanged: the simulation time frame is still divided into sub-intervals, and a global fixed-point strategy is used to converge the meshes and the solutions. On each sub-interval, the mesh number of vertices remains the same. However, and it is the first difference, within a sub-interval, the mesh is moved using the connectivity-change moving mesh algorithm presented in Section 5.1 and fully detailed in [1]. One or several mesh deformation steps are performed during a sub-interval. The connectivity-change moving mesh algorithm uses mesh optimizations to make

the mesh deformation efficient and robust. These mesh optimizations, smoothing and swapping operators, use the dynamic (corrected) metric field presented in Section 5.3 to be compliant with the current dynamic adapted mesh.

The second difference concerns the computation of the ALE Hessian-metric for the next adaptation step. ALE Hessian-metrics $|H_u^{i,\mathrm{ALE}}|$ need to be sampled in time. To compute each sample, we use the displacement given by the mesh deformation step without taking into account the smoothing optimization. In our numerical experiments, we did not see any significant difference on the solution accuracy and the quality of the adapted meshes by considering or not the smoothing in the ALE Hessian-metrics computations. This is due mainly to the fact that the mesh smoothing has only a slight influence in 3D, the mesh displacement is predominantly governed by the mesh deformation [1]. The mesh smoothing is only a slight perturbation around the mesh deformation displacement to get better shaped elements. Nevertheless, we prefer not to consider the smoothing displacement when computing ALE Hessian-metrics because:

- the geometry displacement is clearly defined, thus the mesh deformation is converging toward a unique mesh deformation and therefore we converge toward a unique optimal ALE continuous mesh
- the smoothing correction for one node may change from one fixed-point iteration to the other, thus for each node it does not converge to a fixed correction.

The gradient of the mesh deformation displacement field is computed on the mesh position at the beginning of the sub-interval. The Hessian of the sensor is computed on the mesh current position and is mapped back to the mesh position at the beginning of the sub-interval. Then, as in the standard algorithm, the ALE Hessian-metrics are computed at each vertex.

Note that the connectivity-change moving mesh algorithm allows the mesh vertices to move to their final position given by the mesh deformation solution while it would be impossible to reach such positions without skewing mesh elements with a classical method. Thanks to the robustness of this algorithm, we are always able to calculate the ALE hessian-metrics.

6. Numerical examples

We now present several three-dimensional CFD simulations to illustrate the efficiency of the moving-mesh unsteady fixed-point adaptation algorithm. The presented examples were run on 20 cores using a 2 Xeon E5-2670 v2 chip processor (10 cores at 2.5 GHz for each chip), both chips being connected by 2 QPI links with a speed of 16 GB/s.

6.1. A shock tube in expansion

The first example is a variation of the classic Sod shock-tube problem, with a tube being homogeneously expanded in one direction. This *a priori* simple test case shows that the refined regions follow the physical phenomena of interest, and that the anisotropy is well preserved despite the mesh being moved.

The tube initial dimensions are $[0,1]\times[-0.15,015]\times[0,1]$, and the gas is split in two regions: for $x \leq 0.82$ the state is $(\rho_{\text{left}}, \mathbf{u}_{\text{left}}, p_{\text{left}}) = (1, \mathbf{0}, 1)$ whereas for $x > 0.82$ the state is $(\rho_{\text{right}}, \mathbf{u}_{\text{right}}, p_{\text{right}}) = (0.125, \mathbf{0}, 1)$. The gas is then left free to evolve, and the classic rarefaction wave, contact discontinuity and shock discontinuity appear and evolve in the tube. The tube is expanded to the right with the following movement imposed to the mesh vertices:

$$\begin{cases} x(t) &= x_0 + 1.5\, x_0\, t \\ y(t) &= y_0 \\ z(t) &= z_0\,. \end{cases}$$

The simulation is run until time $T = 0.6$ so that the size of the tube doubles in the x-direction. For this case, 5 fixed-point iterations and 20 sub-intervals were prescribed, and the density field is used as sensor for the adaptation. A space-time complexity $\mathcal{N}_{st} = 80,000$ was prescribed leading to a sub-interval average spatial complexity $\mathcal{N}_{avg} = 4,000$. The resulting adapted meshes at the last fixed-point iteration have an average number of spatial vertices equal to $24,500$. A total of $n_{iter} = 7,075$ time-steps have been performed in the last fixed-point iteration. The total number of space-time vertices at the last fixed-point iteration is $N_{st} = 174$ million. For this simulation, the CPU time to compute the solution at the last fixed-point iteration (flow solver and interpolation steps on the last set of sub-interval adapted meshes) represents 43% of the total CPU time. In other words, the overhead of the iterative process to obtain the optimal adapted space-time mesh is slightly higher than a factor two. This is mainly due to the fact that the accuracy of the mesh is increased at each fixed-point iteration as the mesh-solution convergence is being established.

The meshes at different time-steps and the corresponding solution are shown in Figure 8. Not only do the waves evolve in the refined bands, but these bands also move as the tube is expanded. The adapted regions are transported together with the mesh in accordance with our algorithm, that associates metrics to moving vertices rather than to fixed positions in space. In Figure 9, we

show the mesh at the beginning and at the end of the fourteenth sub-interval, to emphasize the movement of the adapted regions throughout a sub-interval, together with the movement of the tube boundaries, still preserving the anisotropy. In Figure 10, we zoom on the solution at the beginning and the end of the fourteenth sub-interval to stress that the evolving physical phenomena remain inside the adaptation bands during a sub-interval, despite the mesh deformation and the fact that bands themselves are moving. We notice that the region where the shock wave evolves during the sub-interval is a lot larger than the refined band size because the refined band moves forward together with the shock wave. Therefore, the size of refined band has been reduced thanks to the ALE metric.

This case has the advantage of having a known analytic solution, which makes it possible to analyze quantitatively the effects of the adaptation. Three versions of this case are compared: the first one is a moving mesh ALE simulation without adaptation, the second one is an adaptive simulation with no moving mesh, the tube being fixed in expanded position, and the third one is a fully moving mesh adaptive simulation. In all three simulations, the same space-time complexity is prescribed. The results

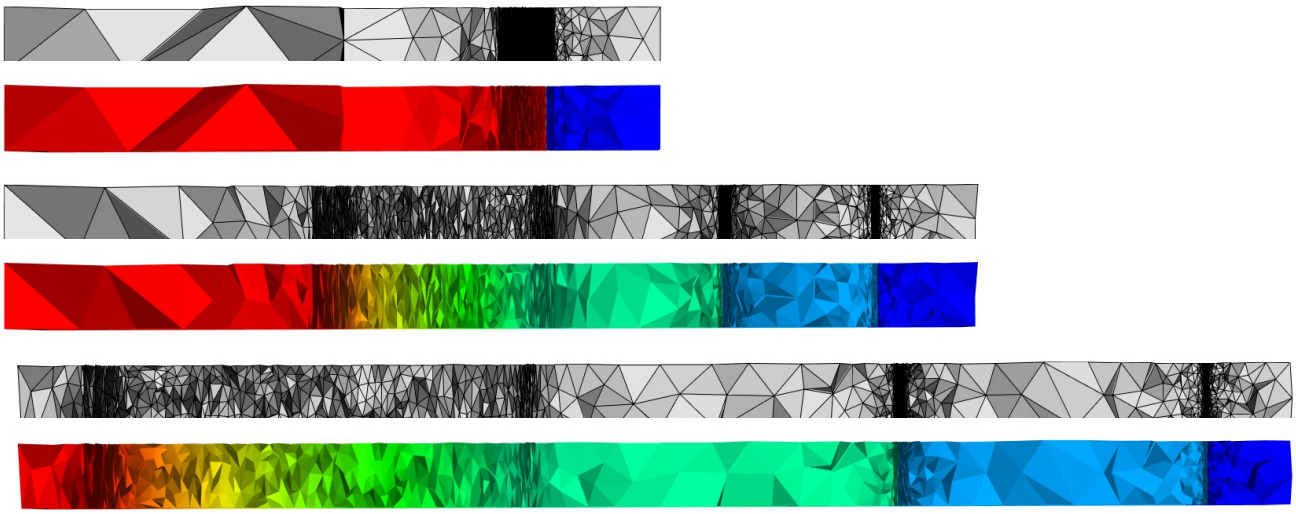

Figure 8: Adapted meshes and density solutions for the expanding shock tube case at time 0, 0.5 and 0.66. Cuts into the volume mesh are made along the plane $y = 0$.

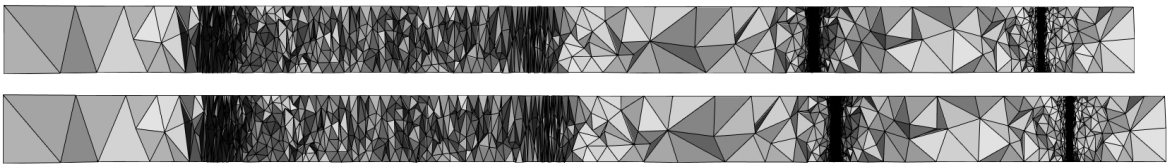

Figure 9: Mesh at the beginning (top) - $t = 0.42$ - and end (bottom) - $t = 0.45$ - of the fourteenth sub-interval. One can see the refined bands are moved forward as the tube is being expanded.

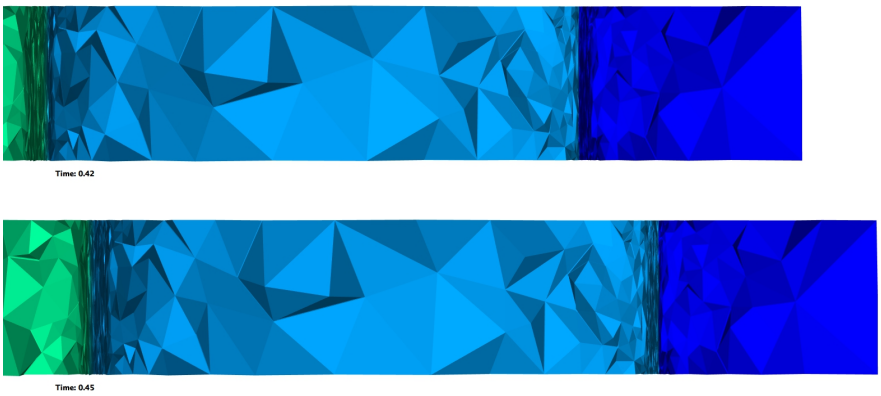

Figure 10: Zoom on the mesh and solution at the beginning (top) - $t = 0.42$ - and at the end (bottom) - $t = 0.45$ - of the fourteenth sub-interval. One can see the two waves move forward inside the adapted bands while the refined bands are moving forward.

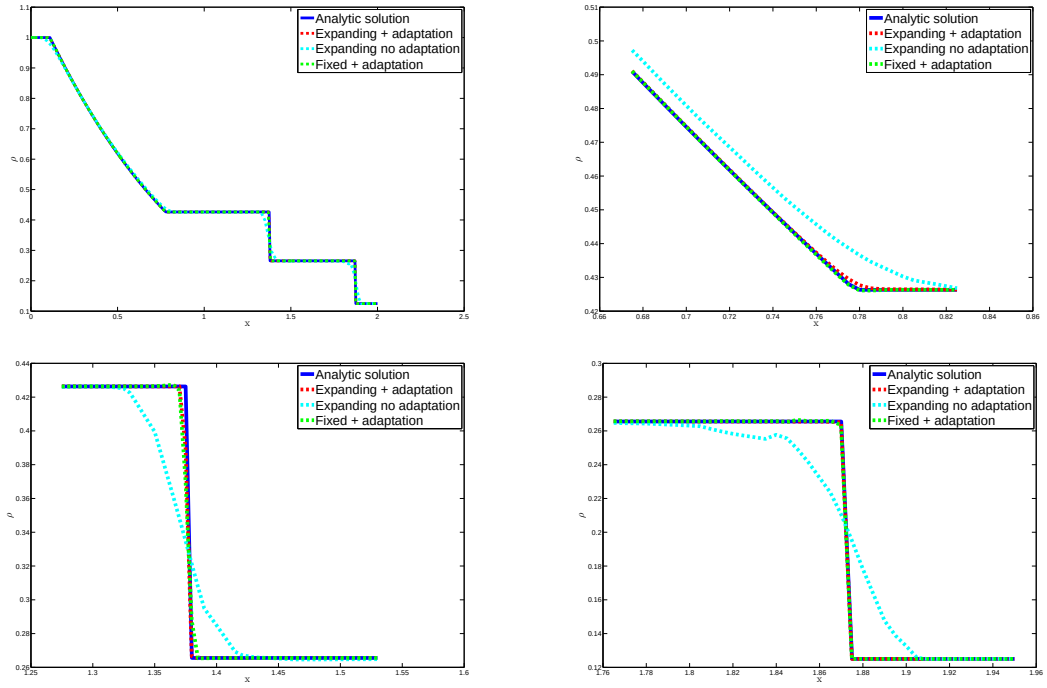

Figure 11: Density solution extraction along a line inside the domain for a moving mesh ALE simulation without adaptation, an adaptive simulation with no moving mesh, and moving mesh adaptive simulation. The extractions are compared to the analytic solution on the whole simulation interval (first graph) and on zooms on the different features.

are presented in Figure 11 and point out that the moving mesh adapted solution matches the analytic solution as well as the fixed mesh adapted solution, both being much more accurate than the non-adapted ALE simulation in the shock, the contact discontinuity and the expansion regions.

To emphasize the efficiency of the proposed approach, a quantitive CPU analysis of this simulation is carried out by comparing the adaptive simulations with non-adaptive simulations using uniform meshes. Then, we quantify the CPU overhead due to moving meshes by comparing the adaptive simulations with and without moving meshes. Please note that in general, one does not run a moving mesh simulation when a fixed mesh simulation is possible.

Adaptive mesh versus uniform mesh simulations CPU efficiency. We compare the CPU time obtained with the global fixed-point mesh adaptation for moving mesh simulations (Algorithm 1 and Section 5.4) and simulations on uniform meshes on this shock tube in expansion problem. The CPU time in each case is analyzed with respect to the solution accuracy at final time $T = 0.6$, *i.e.,* the spatial error in L^1-norm of the solution at time $T = 0.6$ with respect to the analytical solution. For a reliable analysis, the comparison is done for many simulations to observe the convergence of the error with respect to the CPU time. This CPU time analysis was run on 12 cores using a Xeon E5-2697 v2 chip processor at 2.7 GHz.

All parameters of the ALE flow solver are identical for all simulations. The CPU time for the uniform mesh simulations corresponds to the CPU time of the ALE flow solver to run such simulations. Four uniform mesh simulations have been run with a number of vertices ranging from $450,000$ to 3.5 million. The CPU time in the adaptive case corresponds to the sum of all the CPU times for each stage of the adaptive algorithm (flow solver, interpolation, metric fields construction, and remeshing) for all fixed-point iterations. Fifteen adaptive mesh simulations have been run using 6, 12 and 18 sub-intervals and a theoretical average complexity of $1,000$, $2,000$, $4,000$, $6,000$ and $8,000$ leading to sub-interval adapted meshes with a size ranging from $6,420$ to $46,204$ vertices. The convergence in the adaptive case can be either observed by fixing the number of sub-intervals and increasing the theoretical complexity, see Figure 12 (left), or by fixing the theoretical complexity and increasing the number of sub-intervals, see Figure 12 (right). In these plots, we clearly see the superiority of the adaptive simulations. Indeed, the error is reduced by a factor 4 for the more accurate simulations, and we also observe a higher order of convergence for adaptive simulations. To reach the accuracy of the adaptive simulations, the uniform mesh simulations would need a lot more vertices.

A more detailed error analysis has been done in the fixed-boundary case [6] showing the great improvement of the adaptive simulations versus non-adaptive simulations. This work also explains why, in terms of convergence, it is more advantageous to increase the number of sub-intervals[5] while performing a convergence study instead of increasing the average complexity as can be seen in Figure 12.

[5]Changing the number of sub-intervals optimizes (adapts) the space-time mesh in the time direction.

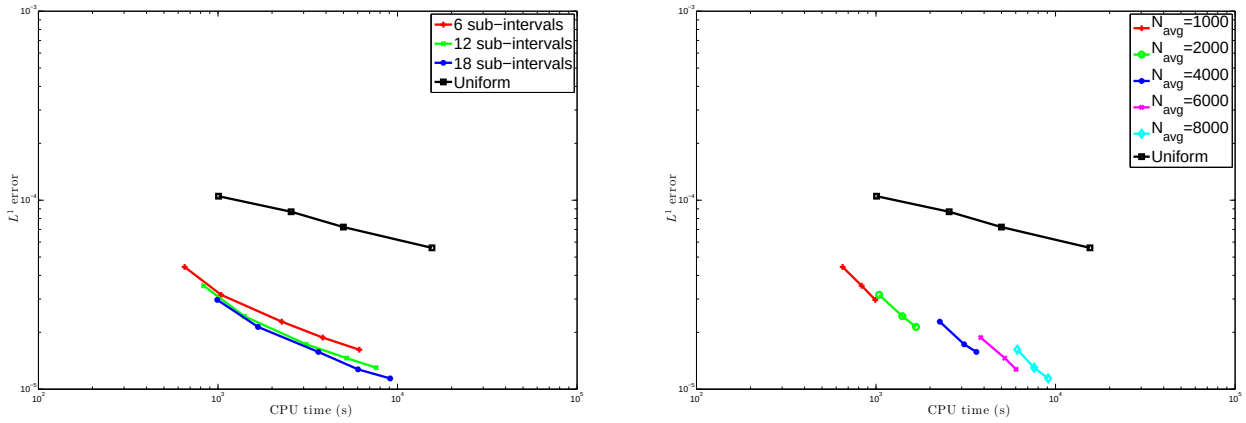

Figure 12: Convergence curves of the spatial error of the final solution with respect to the CPU time for uniform mesh simulations (black) and adaptive mesh simulations (red, green, blue, pink and cyan). The convergence in the adaptive case can be either observed by fixing the number of sub-intervals and increasing the theoretical complexity (left) or by fixing the theoretical complexity and increasing the number of sub-intervals (right).

CPU overhead of moving mesh simulations. We analyze the CPU overhead of the ALE solver with respect to the solver on fixed meshes. This overhead is due to the update of the finite volume cells, the computation of the mesh velocities, and the extra terms in the ALE fluxes. When an explicit time-stepping is considered, if the cost of the solver on fixed mesh is 1 then the cost of the solver on moving meshes is around 1.75. There is also an overhead due to mesh deformation: the elasticity solutions, the mesh optimizations, moving the mesh and checking its validity. For this case, if the cost of the solver on fixed mesh is 1 then the cost of the mesh deformation is 0.2. Moreover, in this very specific case, a total of $7,075$ time-steps have been performed for the moving mesh adaptive simulation whereas $3,721$ time-steps have been performed for the fixed mesh adaptive simulation. Indeed, as the tube is expanding and to preserve the solution accuracy, the ALE metric field prescribes a mesh with a size smaller than required in the x-direction at the beginning of the sub-intervals because this size is going to grow during the sub-intervals. Thus, a larger number of time-steps are performed for the moving mesh adaptive simulation. For the whole simulation, adding all these overheads, the cost of the ALE flow solver is 3.31 times the cost of the flow solver on fixed meshes, see Table 3.

We can then analyze the global cost of the adaptive simulations. In Table 3, we observe that there is no overhead for the interpolation or the remeshing parts. Thus, the adaptive moving mesh case is 2.85 times the cost of the adaptive fixed mesh case. The percentage of the total CPU time for the different stages of the fixed-point mesh adaptation algorithm for both cases are given in Table 4. It is clear that for these cases most of the CPU time is spent in the flow solver.

	Interpolation CPU	Flow Solver CPU	Metric & Remeshing CPU	Total
Fixed mesh case	0.07	0.80	0.13	1
Moving mesh case	0.07	2.65	0.13	2.85

Table 3: Sod shock-tube problem. CPU ratio with respect to the total CPU time of the adaptive fixed mesh simulation for the different stages of the fixed-point mesh adaptation algorithm, for the fixed mesh and for the moving mesh cases.

	Interpolation CPU %	Flow Solver CPU %	Metric & Remeshing CPU %
Fixed mesh case	6.96%	79.97%	13.07%
Moving mesh case	2.57%	92.78%	4.65%

Table 4: Sod shock-tube problem. Percentage of the total CPU time for the different stages of the fixed-point mesh adaptation algorithm, for the fixed mesh and the moving mesh cases.

6.2. A moving ball in a shock tube

In the second example, we consider again a Sod shock tube, but instead of expanding the tube, we add a moving ball inside it, that interacts with the shock and the contact discontinuity.

The dimensions of the tube are $[0,1]\times[-0.15,0.15]\times[-0.1,0.1]$. At initial time, the gas is split in two states: for $x \leq 0.5$ the state is $(\rho_{\text{left}}, \mathbf{u}_{\text{left}}, p_{\text{left}}) = (1, \mathbf{0}, 1)$ whereas for $x > 0.5$ the state is $(\rho_{\text{right}}, \mathbf{u}_{\text{right}}, p_{\text{right}}) = (0.125, \mathbf{0}, 1)$. A ball of radius $r = 0.02$ is immersed in the gas, its center being in position $(0.75, 0, 0)$. The tube is fixed, while the ball has a constant speed $\mathbf{v}_{ball} = -0.3\,\mathbf{e}_{\mathbf{x}}$. The simulation is run until final time $T = 0.5$. After a while, the ball goes through the shock and the contact discontinuity,

creating complex patterns both in front of it and in its wake. We expect all these physical phenomena to be captured correctly by the multiscale adaptation algorithm. For this case, 80 sub-intervals and 5 fixed-point iterations were prescribed, and the density field is used as sensor for the adaptation. A space-time complexity of 12 million was prescribed leading to a sub-interval average spatial complexity of $\mathcal{N}_{avg} = 150,000$. The resulting adapted meshes at the last fixed-point iteration have an average number of spatial vertices equal to $265,000$ with sizes ranging from $100,000$ to $450,000$ vertices. A total of $n_{iter} = 1,854$ time-steps have been performed in the last fixed-point iteration. The total number of space-time vertices N_{st} at the last fixed-point iteration is 500 million. For this simulation, the CPU time to compute the solution at the last fixed-point iteration (flow solver and interpolation steps on the last set of sub-interval adapted meshes) represents 20% of the total CPU time.

The adapted meshes at different time-steps and the corresponding solutions are shown in Figure 13. Since we are interested in the interaction of the ball with the shock and the contact discontinuity, we only show the part $x > 0.5$ of the tube. The meshes shown are the ones at the end of the sub-intervals, *i.e.,* the meshes that have been moved. Thanks to the anisotropic mesh adaptation, a highly resolved solution is obtained: the shock wave and the contact discontinuity are accurately captured all along

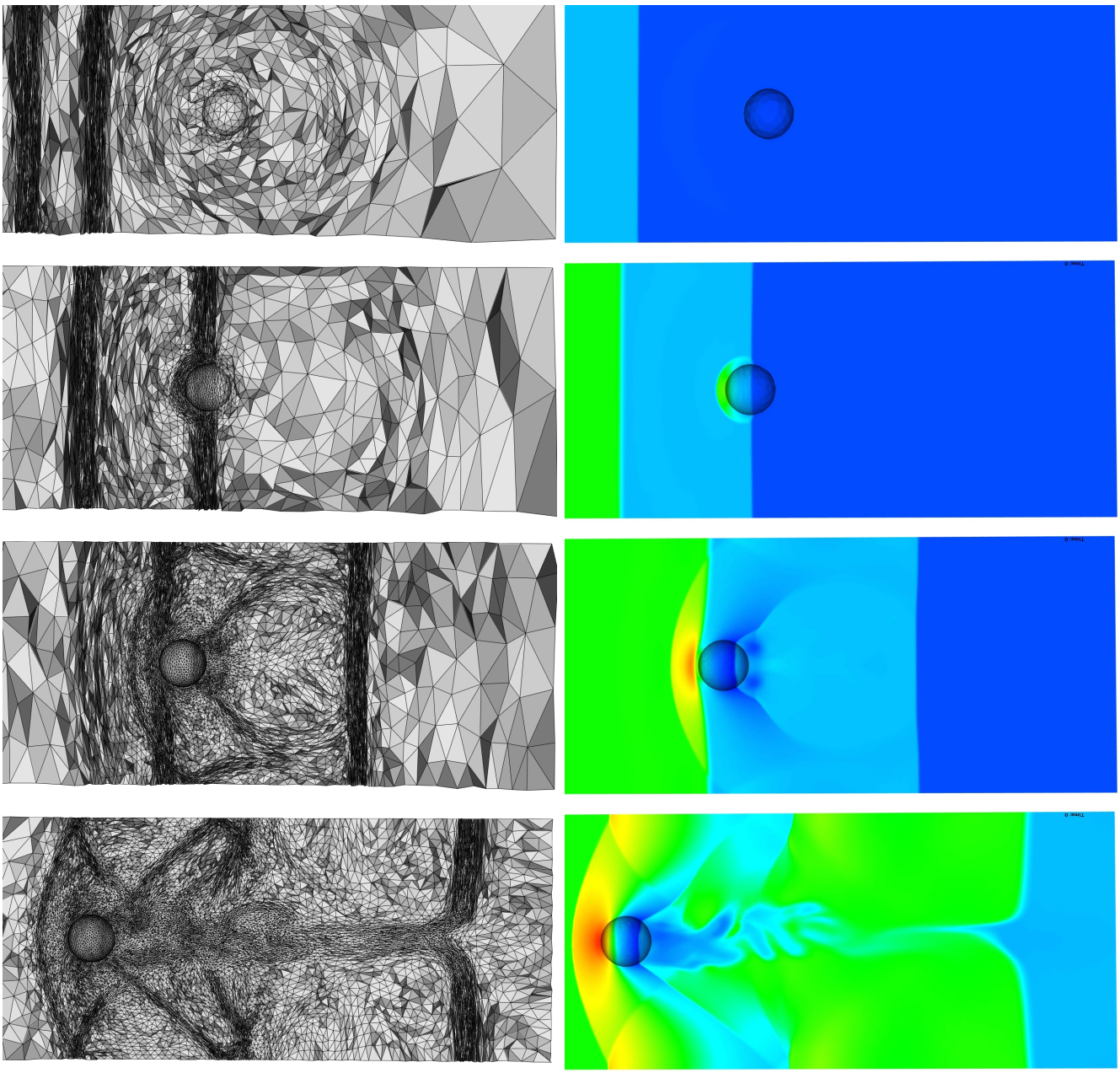

Figure 13: Adapted meshes and density solutions for the moving ball in a shock tube test case, at times 0.07, 0.12, 0.19 and 0.44. Cuts into the volume mesh are made along the plane $y = 0$.

the simulation, and more complex features around the ball and in its wake are represented in detail. It is interesting to note that the shock is perfectly meshed as soon as the ball has gone through it. A closer look at the surface of the ball also shows that it is adapted with highly anisotropic elements, and the anisotropy close to the surface is well preserved.

For this simulation, allowing mesh optimization (smoothing and edge/face swapping) is mandatory, otherwise, the shearing created by the advance of the ball in the volume mesh quickly creates invalid elements. Consequently, it is crucial to compute the quality criteria that trigger optimization with regards to the actual metric of the mesh. In this case, it is clear in figures shown that the connectivity-change optimization did not break the anisotropy in the adapted regions, although nearly 4.5 millions of swaps were performed during the last fixed-point iteration, *i.e.,* on average $55,000$ swaps per sub-interval are performed. To give an idea of the impact of the swaps on the numerical scheme, it represents nearly $2,500$ swaps per time-step as $1,854$ time-steps have been performed at the last fixed-point iteration. These $2,500$ swaps have to be compared to the number of elements in the meshes which is 1.6 millions on average for each sub-interval, *i.e.,* one swap for 640 elements per time-step. The CPU distribution between the different stages of the fixed-point mesh adaptation algorithm are given in Table 5.

6.3. Nosing-up F117 aircraft

The third example is a subsonic notional F117 aircraft geometry nosing up, that creates a vortical wake. An inflow of air at Mach 0.4 arrives in front of the aircraft, initially in horizontal position, that noses up, stays up for a while, then noses down. In this example, the aircraft rotates around its center of gravity. Let $T = 1s$ be the characteristic time of the movement and $\theta^{max} = 20^o$ the maximal angle reached, the movement is defined by its angle of rotation, of which the evolution is divided in 7 phases:

$$\theta(t) = \theta^{max} \begin{cases} 0 & \text{if} \quad 0 \le t \le T/2 \qquad (i) \\ \frac{2(t-\frac{T}{2})}{T^2} & \text{if} \quad \frac{T}{2} < t \le T \qquad (ii) \\ \frac{1}{2} + 2\,(\frac{2\,(t-\frac{T}{2})}{T} - 1) - \frac{1}{2}\,(\frac{4\,(t-\frac{T}{2})^2}{T^2} - 1) & \text{if} \quad T < t \le \frac{3T}{2} \qquad (iii) \\ 1 & \text{if} \quad \frac{3T}{2} < t \le \frac{7T}{2} \qquad (iv) \\ 1 - \frac{2\,(t-\frac{7T}{2})}{T} & \text{if} \quad \frac{7T}{2} < t \le 4T \qquad (v) \\ \frac{1}{2} - 2\,(\frac{2\,(t-\frac{7T}{2})}{T} - 1) + \frac{1}{2}\,(\frac{4\,(t-\frac{7T}{2})^2}{T^2} - 1) & \text{if} \quad 4T < t \le \frac{9T}{2} \qquad (vi) \\ 0 & \text{if} \quad \frac{9T}{2} t \le 5T \qquad (vii) \end{cases}$$

Phase (i) is an initialization phase, during which the flow around the aircraft is established. Phases (ii) and (iii) are respectively phases of accelerated and decelerated ascension. Vortices start to grow behind the aircraft, and they expand during phase (iv), where the aircraft stays in upward position. Phases (v) and (vi) are phases of accelerated and decelerated descent, the vortices start to move away and they slowly disappear in phase (vii). Free-stream conditions are imposed on the faces of the surrounding box, and slipping conditions on the aircraft.

The time simulation interval $[0, 5]$ was divided into 96 sub-intervals, and 5 fixed-point iterations are performed. The adaptation sensor is the local Mach number. A space-time complexity of 48 million was prescribed leading to a sub-interval average spatial complexity $\mathcal{N}_{avg} = 500,000$. The adapted meshes at the last fixed-point iteration have between $100,000$ and $1,400,000$ vertices, for an average number of vertices equal to $841,000$. A total of $n_{iter} = 38,980$ time-steps have been performed in the last fixed-point iteration. The total number of space-time vertices N_{st} at the last fixed-point iteration is equal to 35 billion. The total CPU time of this simulation is 95 hours, 31 hours being spent to compute the solution in the last fixed-point iteration (flow solver and interpolation steps on the last set of sub-interval adapted meshes) which represents 32% of the total CPU time. The CPU distribution between the different stages of the fixed-point mesh adaptation algorithm are given in Table 5.

Views of the adapted meshes and the corresponding solutions are displayed in Fig. 14. The vortical wake is propagated far from the aircraft, and the patterns of the vortices are highly resolved. The observation of the adapted meshes shows that they are actually refined only in the vicinity of the wake, and with such a precision that one can see the vortices being created, evolving and vanishing just by looking at the meshes. A view at several meshes within the same sub-interval shows that the mesh evolves continuously following the physical phenomena. In the wake far from the aircraft, the elements are highly anisotropic, whereas they have to be isotropic in the area of vorticity due to the characteristic of the physical phenomena.

6.4. Two F117 aircraft flight paths crossing

This case is an example of moving mesh simulation, described in [11]. It models two notional F117 aircraft geometries having crossing flight paths, translating and rotating. This problem is difficult in terms of mesh movement and it illustrates the efficiency of the connectivity-change moving mesh algorithm in handling large displacements of complex geometries without any remeshing. When both aircraft cross each other, the mesh encounters large shearing due to the opposite flight directions. The connectivity-change mesh deformation algorithm handles easily this complex displacement thanks to the mesh local reconnections.

Concerning the fluid simulation, the aircraft are moved at a speed of Mach 0.4, in an initially inert uniform fluid: at $t = 0$ the speed of the air is zero everywhere. Transmitting boundary conditions are used on the sides of the surrounding box, while slipping conditions are imposed on the two F117 bodies. After a short phase of initialization, the flow is established when the two F117s pass each other, and the density fields around the aircraft and in their wake interact. Acoustic waves are created in front of the F117s due to the instantaneous setting in motion of the aircraft.

The adaptation parameters are the following: the adaptation is performed on the density field, the time interval is divided into 50 sub-intervals, and 4 fixed-point iterations are performed. A space-time complexity 20 million was prescribed leading to a sub-interval average spatial complexity $\mathcal{N}_{avg} = 400,000$. The adapted meshes at the last fixed-point iteration have between $360,000$ and $820,000$ vertices, for an average number of vertices equal to $745,000$. A total of $n_{iter} = 93,875$ time-steps have been performed in the last fixed-point iteration. The total number of space-time vertices N_{st} at the last fixed-point iteration is equal to 70 billion. The total CPU time of this simulation is 130 hours, 58 hours being spent to compute the solution in the last

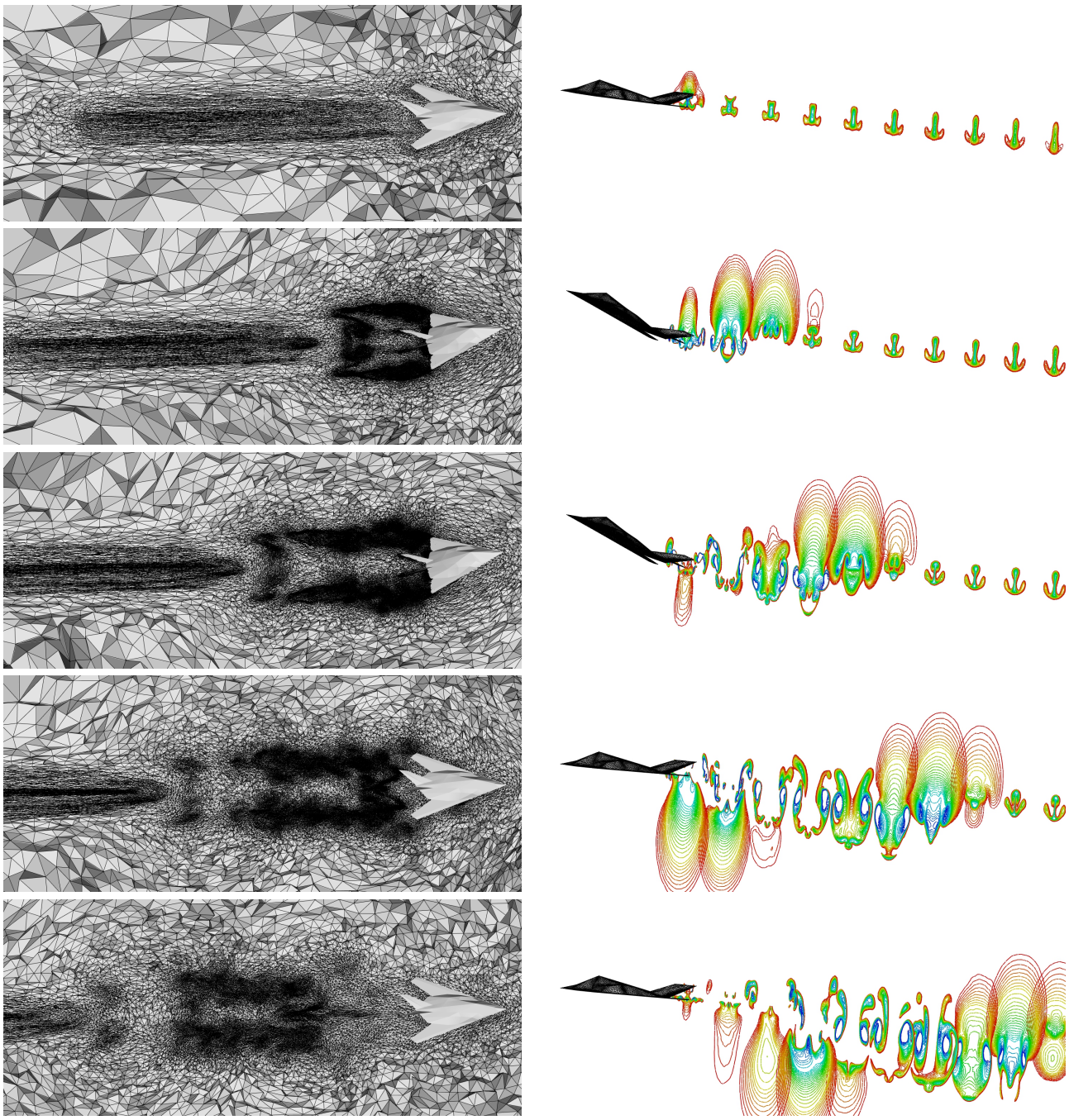

Figure 14: Nosing-up F117 test case: adapted meshes (view from the top) and local Mach number isolines at different time steps.

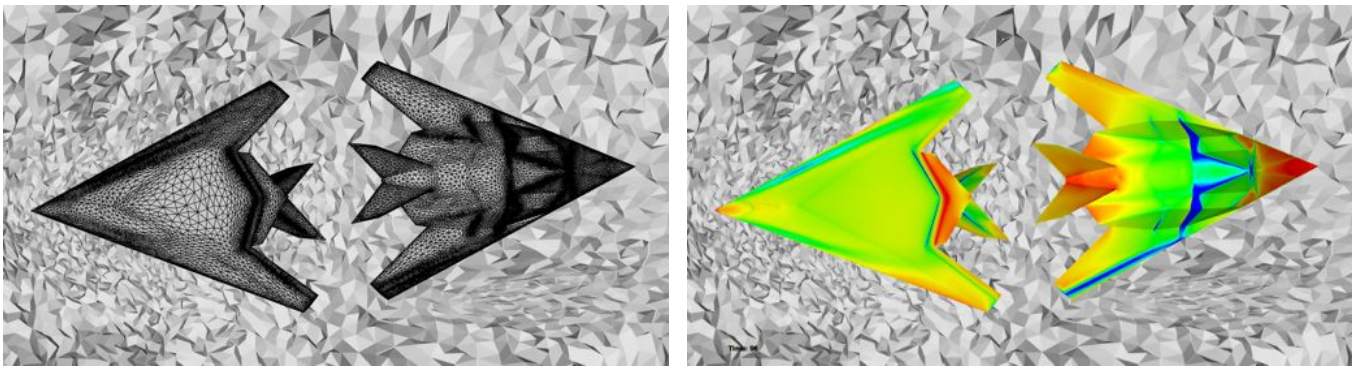

Figure 15: Close-ups on the surface mesh and solution of the two F117s.

fixed-point iteration (flow solver and interpolation steps on the last set of sub-interval adapted meshes) which represents 44% of the total CPU time. The CPU distribution between the different stages of the fixed-point mesh adaptation algorithm are given in Table 5.

Note that this simulation involves only 50 remeshing and solution interpolation between meshes at each fixed-point iteration. If we had considered a strategy where the mesh is frequently remeshed, for instance each 10 time-steps, the generation of $9,387$ adapted meshes and $9,387$ solution interpolation between meshes would have been required. It is clear that such a strategy would have been a lot less efficient and a lot less accurate that the method proposed in this paper.

The meshes and density solutions at different time-steps are shown in Figure 16. The meshes are well adapted to the solution. Thanks to the adaptation, the wakes of the F117s are captured far from the aircraft, and their interactions with the wakes are clearly visible. The anisotropy of the meshes is obvious, in front of the aircraft (corresponding to the acoustic waves), and in their wake. In Figure 15, a close up on the two aircraft is made, that shows the adapted surface mesh and solution. In this case again, a correct handling of the mesh optimization with a dynamic ALE metric is necessary to preserve the anisotropic mesh adaptation.

One can conclude from these examples results that the adaptive connectivity-change moving mesh algorithm has proved to be very efficient in deforming anisotropic adapted meshes containing anisotropic elements with high aspect ratio when large geometry displacements are involved.

	Interpolation CPU	Flow Solver CPU	Metric & Remeshing CPU
Moving ball	10.80%	79.22%	11.04%
F117 nosing-up	4.94%	89.84%	5.73%
Two F117 crossing	0.89%	96.73%	2.37%

Table 5: Percentage of the total CPU time for the different stages of the fixed-point mesh adaptation algorithm on the last three examples.

7. Conclusion

This paper has addressed time-accurate anisotropic mesh adaptation in the case of dynamic meshes within the range of body-fitted ALE simulations. The multiscale space-time interpolation error analysis for time-dependent problems has been extended to the case of dynamic meshes leading to the expression of the optimal ALE continuous mesh. The mesh deformation is taken into account in this expression: the ALE metric field is used to generate adapted meshes that will remain adapted once moved as required by the geometry movements. Then, the global fixed-point unsteady mesh adaptation algorithm has been coupled to a connectivity-change moving mesh algorithm. The ideas driving this algorithm remain the same, and only few modifications are required: use the connectivity-change moving mesh algorithm to move the vertices, compute and use the ALE metric field and use a dynamic ALE metric field that takes into account the mesh deformation for the mesh optimization.

Several numerical examples were presented in three dimensions, that validate the proposed approach. Adapted regions remain around the solution flow features, even if the mesh vertices are moved in other directions to follow the moving boundaries, and we have successfully managed to avoid the frequent remeshings usually performed for this kind of simulations. This results in a better accuracy at a lesser cost. Indeed, we have shown the superiority in CPU time efficiency of the adaptive algorithm with respect to uniform mesh simulations. These examples show both that our theoretical optimal ALE continuous mesh effectively

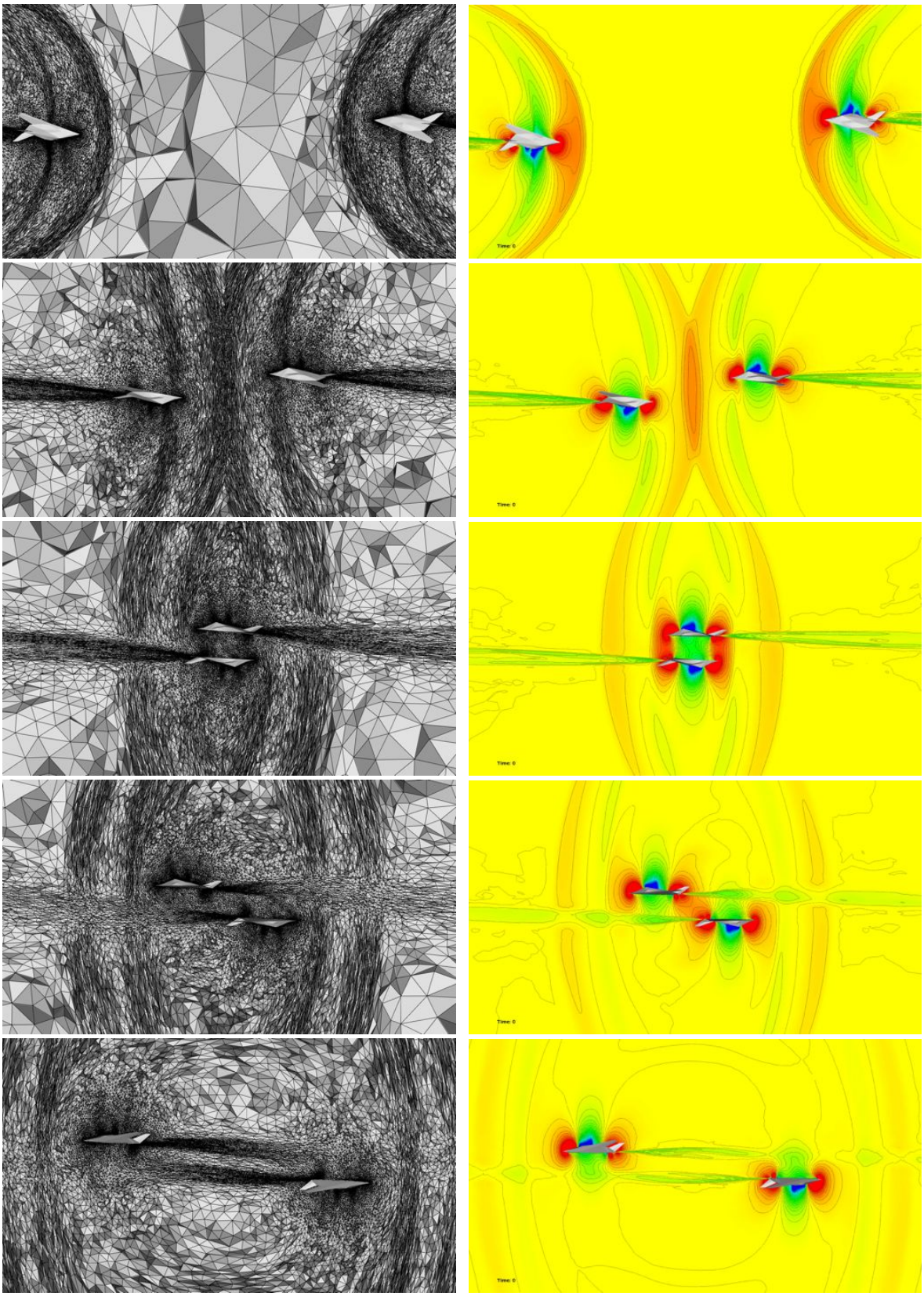

Figure 16: Adapted meshes and density solutions for the two F117s test case at different time-steps. Cuts into the volume mesh are made along the plane $y = 0$.

leads to highly resolved solutions, and that it can be put into practice at a reasonable time cost for complex fluids problems: in 3D, with complex boundaries undergoing large displacements.

Several improvement tracks are considered. First, it is now within our reach to extend goal-oriented unsteady error estimates of [13] to moving boundary problems. Second, the mesh deformation due to the moving geometry may lead us to over- refine some regions of the computational domain. For instance, if we consider an expanded tube that grows by a factor n on a sub-interval, if an element size of h is expected at the end of the sub-interval, we may have to generate an element size of h/n at the beginning of the sub-interval. This issue is recurrent, for instance in piston engine simulations. In that case, we may want to adapt the sub-interval time length Δt in order to control this over-refinement by a given factor. This will also provide a control on the time step. Finally, this paper has only discussed the case of an explicit temporal scheme. If an implicit scheme is considered with possibly arbitrary large time-steps to increase the flow solver efficiency, the size of the time-steps should be adapted in order to control the error introduced by the temporal scheme. To this end, the time step has to be introduced in the space-time error analysis. A first work in this direction has been proposed in [19].

8. Acknowledgments

This work was partially funded by the Airbus Group Foundation and was partly done in the MAIDESC ANR project which is supported by the french ministry of Research under contract ANR-13-MONU-0010.

References

[1] F. Alauzet. A changing-topology moving mesh technique for large displacement. *Eng. w. Comp.*, 30(2):175–200, 2014.
[2] F. Alauzet. A parallel matrix-free conservative solution interpolation on unstructured tetrahedral meshes. *Comput. Methods Appl. Mech. Engrg.*, 299:116–142, 2016.
[3] F. Alauzet, P.J. Frey, P.L. George, and B. Mohammadi. 3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations. *J. Comp. Phys.*, 222:592–623, 2007.
[4] F. Alauzet and A. Loseille. On the use of space filling curves for parallel anisotropic mesh adaptation. In *Proceedings of the 18th International Meshing Roundtable*, pages 337–357. Springer, 2009.
[5] F. Alauzet and A. Loseille. High order sonic boom modeling by adaptive methods. *J. Comp. Phys.*, 229:561–593, 2010.
[6] F. Alauzet, A. Loseille, and G. Olivier. Multi-scale anisotropic mesh adaptation for time-dependent problems. RR-8929, INRIA, June 2016.
[7] F. Alauzet and G. Olivier. Extension of metric-based anisotropic mesh adaptation to time-dependent problems involving moving geometries. In *49th AIAA Aerospace Sciences Meeting*, AIAA Paper 2011-0896, Orlando, FL, USA, Jan 2011.
[8] L. Arpaia and M. Ricchiuto. Mesh adaptation by continuous deformation. basics: accuracy, efficiency, well balancedness. RR-8666, INRIA, January 2015.
[9] N. Barral. *Time-accurate anisotropic mesh adaptation for three-dimensional moving mesh problems*. PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2015.
[10] N. Barral and F. Alauzet. Large displacement body-fitted FSI simulations using a mesh-connectivity-change moving mesh strategy. In *44th AIAA Fluid Dynamics Conference*, AIAA Paper 2014-2773, Atlanta, GA, USA, Jun 2014.
[11] N. Barral, E. Luke, and F. Alauzet. Two mesh deformation methods coupled with a changing-connectivity moving mesh method for CFD applications. In *Proceedings of the 23th International Meshing Roundtable*. Elsevier, 2014.
[12] J.D. Baum, H. Luo, and R. Löhner. A new ALE adaptive unstructured methodology for the simulation of moving bodies. In *32th AIAA Aerospace Sciences Meeting*, AIAA Paper 1994-0414, Reno, NV, USA, Jan 1994.
[13] A. Belme, A. Dervieux, and F. Alauzet. Time accurate anisotropic goal-oriented mesh adaptation for unsteady flows. *J. Comp. Phys.*, 231:6323–6348, 2012.
[14] Y. Bourgault and M. Picasso. Anisotropic error estimates and space adaptivity for a semidiscrete finite element approximation of the transient transport equation. *SIAM J. Sci. Comput.*, 35(2):1192–1211, 2013.
[15] P.A. Browne, C.J. Budd, C. Piccolo, and M. Cullen. Fast three dimensional r-adaptive mesh redistribution. *J. Comp. Phys.*, 275:174–196, 2014.
[16] J. Carpio and J.L. Prieto. An anisotropic, fully adaptive algorithm for the solution of convection-dominated equations with semi-Lagrangian schemes. *Comput. Methods Appl. Mech. Engrg.*, 273:77–99, 2014.
[17] L. Chacón, G.L. Delzanno, and J.M. Finn. Robust, multidimensional mesh-motion based on Monge-Kantorovich equidistribution. *J. Comp. Phys.*, 230(1):87–103, 2011.
[18] G. Compere, J-F. Remacle, J. Jansson, and J. Hoffman. A mesh adaptation framework for dealing with large deforming meshes. *Int. J. Numer. Meth. Engng*, 82(7):843–867, 2010.
[19] T. Coupez, G. Jannoun, N. Nassif, H.C. Nguyen, H. Digonnet, and E. Hachem. Adaptive time-step with anisotropic meshing for incompressible flows. *J. Comp. Phys.*, 241:195–211, 2013.
[20] P.A. de Sampaio, P.R. Lyra, K. Morgan, and N. Weatherill. Petrov-Galerkin solutions of the incompressible Navier-Stokes equations in primitive variables with adaptive remeshing. *Comput. Methods Appl. Mech. Engrg.*, 106:143–178, 1993.
[21] C. Dobrzynski and P.J. Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. In *Proceedings of the 17th International Meshing Roundtable*, pages 177–194. Springer, 2008.
[22] V. Ducrot and P.J. Frey. Anisotropic level set adaptation for accurate interface capturing. In *Proceedings of the 17th International Meshing Roundtable*, pages 159–176. Springer, 2008.
[23] P.J. Frey and P.L. George. *Mesh generation. Application to finite elements*. ISTE Ltd and John Wiley & Sons, 2nd edition, 2008.
[24] P.L. George, F. Hecht, and M.G. Vallet. Creation of internal points in Voronoi's type method. Control and adaptation. *Adv. Eng. Software*, 13(5-6):303–312, 1991.
[25] D. Guégan, O. Allain, A. Dervieux, and F. Alauzet. An L^∞-L^p mesh adaptive method for computing unsteady bi-fluid flows. *Int. J. Numer. Meth. Engng*, 84(11):1376–1406, 2010.
[26] O. Hassan, K.A. Sørensen, K. Morgan, and N. P. Weatherill. A method for time accurate turbulent compressible fluid flow simulation with moving boundary components employing local remeshing. *Int. J. Numer. Meth. Fluids*, 53(8):1243–1266, 2007.

[27] A. Hay and M. Visonneau. Computation of free-surface flows with local mesh adaptation. *Int. J. Numer. Meth. Fluids*, 49:785–816, 2005.
[28] W. Huang and R. D. Russell. Adaptive mesh movement - the MMPDE approach and its applications. *Journal of Computational and Applied Mathematics*, 128(1-2):383–398, 2001.
[29] R. Löhner. Adaptive remeshing for transient problems. *Comput. Methods Appl. Mech. Engrg.*, 75(1-3):195–214, 1989.
[30] R. Löhner. Three-dimensional fluid-structure interaction using a finite element solver and adaptive remeshing. *Computing Systems in Engineering*, 1(2-4):257–272, 1990.
[31] R. Löhner and J.D. Baum. Numerical simulation of a shock interaction with complex geometry three-dimensional structures using a new adaptive H-refinement scheme on unstructured grids. In *28th AIAA Aerospace Sciences Meeting*, AIAA Paper 1990-0700, Reno, NV, USA, Jan 1990.
[32] R. Löhner and J.D. Baum. Three-dimensional store separation using a finite element solver and adaptive remeshing. In *29th AIAA Aerospace Sciences Meeting*, AIAA Paper 1991-0602, Reno, NV, USA, Jan 1991.
[33] R. Löhner and J.D. Baum. Adaptive h-refinement on 3D unstructured grids for transient problems. *Int. J. Numer. Meth. Fluids*, 14(12):1407–1419, 1992.
[34] R. Löhner, J.D. Baum, E. Mestreau, D. Sharov, C. Charman, and D. Pelessone. Adaptive embedded unstructured grid methods. *Int. J. Numer. Meth. Engng*, 60:641–660, 2004.
[35] A. Loseille and F. Alauzet. Continuous mesh framework. Part I: well-posed continuous interpolation error. *SIAM J. Numer. Anal.*, 49(1):38–60, 2011.
[36] A. Loseille and F. Alauzet. Continuous mesh framework. Part II: validations and applications. *SIAM J. Numer. Anal.*, 49(1):61–86, 2011.
[37] A. Loseille, A. Dervieux, and F. Alauzet. Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations. *J. Comp. Phys.*, 229:2866–2897, 2010.
[38] A. Loseille, A. Dervieux, P.J. Frey, and F. Alauzet. Achievement of global second-order mesh convergence for discontinuous flows with adapted unstructured meshes. In *37th AIAA Fluid Dynamics Conference*, AIAA Paper 2007-4186, Miami, FL, USA, Jun 2007.
[39] A. Loseille and R. Löhner. Cavity-based operators for mesh adaptation. *51th AIAA Aerospace Sciences Meeting*, Jan 2013.
[40] R. Loubère, P.-H. Maire, M. Shashkov, J. Breil, and S. Galera. ReALE: A reconnection-based arbitrary-Lagrangian-Eulerian method. *J. Comp. Phys.*, 229(12):4724–4761, 2010.
[41] W.G. Habashi M. Najafiyazdi and M. Fossati. Improved transient-fixed-point mesh adaptation using orthogonality-preserving metric intersection. In *20th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2011-3691, Honolulu, HI, USA, Jun 2011.
[42] D.J. Mavriplis. Revisiting the least-square procedure for gradient reconstruction on unstructured meshes. In *16th AIAA Computational Fluid Dynamics Conference*, AIAA-2003-3986, Orlando, FL, USA, Jun 2003.
[43] S. Murman, M. Aftosmis, and M. Berger. Simulation of 6-DOF motion with cartesian method. In *41th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-2003-1246, Reno, NV, USA, Jan 2003.
[44] G. Olivier. *Anisotropic metric-based mesh adaptation for unsteady CFD simulations involving moving geometries*. PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2011.
[45] G. Olivier and F. Alauzet. A new changing-topology ALE scheme for moving mesh unsteady simulations. In *49th AIAA Aerospace Sciences Meeting*, AIAA Paper 2011-0474, Orlando, FL, USA, Jan 2011.
[46] C.C Pain, A.P. Humpleby, C.R.E. de Oliveira, and A.J.H. Goddard. Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. *Comput. Methods Appl. Mech. Engrg.*, 190:3771–3796, 2001.
[47] C.S. Peskin. Flow patterns around heart valves: a numerical method. *J. Comp. Phys.*, 10:252–271, 1972.
[48] G.M. Porta, S. Perotto, and F. Ballio. A space-time adaptation scheme for unsteady shallow water problems. *Mathematics and Computers in Simulation*, 82:2929–2950, 2012.
[49] R.D. Rausch, J.T. Batina, and H.T.Y. Yang. Spatial adaptation procedures on tetrahedral meshes for unsteady aerodynamic flow calculations. *AIAA Journal*, 30:1243–1251, 1992.
[50] J.-F. Remacle, X. Li, M.S. Shephard, and J.E. Flaherty. Anisotropic adaptive simulation of transient flows using discontinuous Galerkin methods. *Int. J. Numer. Meth. Engng*, 62:899–923, 2005.
[51] P.H. Saksono, W.G. Dettmer, and D. Perić. An adaptive remeshing strategy for flows with moving boundaries and fluid-structure interaction. *Int. J. Numer. Meth. Engng*, 71(9):1009–1050, 2007.
[52] W. Speares and M. Berzins. A 3D unstructured mesh adaptation algorithm for time-dependent shock-dominated problems. *Int. J. Numer. Meth. Fluids*, 25:81–104, 1997.
[53] R.J. Spiteri and S.J. Ruuth. A new class of optimal high-order strong-stability-preserving time discretization methods. *SIAM J. Numer. Anal.*, 40(2):469–491, 2002.
[54] J. Wu, J.Z. Zhu, J. Szmelter, and O.C. Zienkiewicz. Error estimation and adaptivity in Navier-Stokes incompressible flows. *Computational Mechanics*, 6:259–270, 1990.